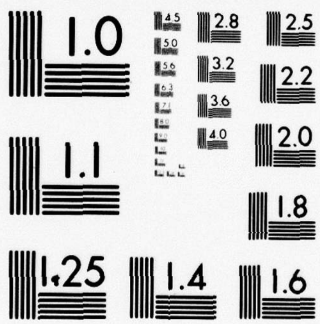
AD-A073 744 STANFORD UNIV CALIF DEPT OF OPERATIONS RESEARCH F/G 12/1
TRANSIENT EFFECTS IN M/G/1 QUEUES: AN EMPIRICAL INVESTIGATION.(U)
JUN 79 M R MIDDLETON N00014-76-C-0418
UNCLASSIFIED TR-85 NL

1 OF 2
ADA
073744



MICROCOPY RESOLUTION TEST CHART

15

6

TRANSIENT EFFECTS IN M/G/1 QUEUES:
AN EMPIRICAL INVESTIGATION.

by

10 MICHAEL R. MIDDLETON

14

9 TECHNICAL REPORT NO. 85, TR-85, TR-51

11 JUNE 1979

12 154 p.

15 PREPARED UNDER CONTRACT
N00014-76-C-0418, (NR-047-061)
NSF-ENG-75-14847
FOR THE OFFICE OF NAVAL RESEARCH

Frederick S. Hillier, Project Director

This research was supported in part by National Science Foundation
Grant ENG 75-14847 Department of Operations Research, Stanford University
issued as Technical Report No. 51

DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA

402 766

JB

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Objective of the Dissertation

The objective of this dissertation is to provide tables for time-dependent expected server load in M/G/1 queueing systems.[1] The results are presented in a form that can be applied by practitioners involved in the design and control of operating systems. This presentation attempts to bridge the gap between the mathematical theory of stochastic processes and the numerical results useful to the management science practitioner.

Waiting lines and congestion are common problems encountered in almost everyone's daily life. A queue can develop in any service system whenever the immediate demand exceeds the system's capacity. One of the problems in designing or controlling such systems is achieving the proper balance between the level of service and the amount of waiting which might occur. Formal analysis may be appropriate if the service involves very expensive equipment, if the costs of waiting are extremely high, or if the decision involves many similar operating systems where the combined costs of service or waiting could be excessive.

The mathematical theory of queues can often provide some insight into the behavior of an actual or proposed operating system. Ideally,

---

[1] This notation is the standard Kendall designation for queueing systems: Markovian (or Poisson) arrivals/General service time distribution/1 (single) server.

the designer would prefer a mathematical model or algorithm that would determine the optimal system design, given the demand or arrival pattern, the costs of various service levels, and the costs associated with customer waiting or delay times. Unfortunately, there is no general optimization technique for queueing models. However, there are numerous predictive models that allow an analyst to determine some operating characteristics of a specific system. Prospective levels of service can be analyzed one at a time and an evaluative model can determine the total cost of service and waiting for each alternative.

The mathematical formulas for such operating characteristics that are often available to the practitioner apply only to queueing systems that are in statistical equilibrium. These steady state results are appropriate for a system where the server can, on the average, serve customers faster than they arrive and where sufficient time has passed to cancel the effects of the system's initial condition. If customers arrive at the system faster than the server can handle them, or if the behavior of the system must be analyzed for the start-up period, then the time-dependent (i.e., transient) operating characteristics are important. Lee [1966, p. 26], in an opinion probably shared by many practitioners, has stated "... the first working rule of queueing theory: time-dependent solutions to queueing-models are either unobtainable or unmanageable."

Transient solutions that are available in the queueing literature are usually expressed in terms of the Laplace transform of the operating characteristic. Bhat [1969, p. B284] reiterates the problem and mentions a possible solution:

"Plainly speaking, the results, given in terms of transforms, very often with more than one argument, fail to make sense to an applied researcher. Numerical inversion of transforms ... is an answer to this problem. But at this stage, the inversion methods are either not sophisticated enough to handle the more complex situations or do not appeal to the applied researcher."

The present research uses an accurate transform inversion technique to produce numerical results for transient expected system load. A practitioner can use these tables to analyze a wide range of M/G/1 systems by referring directly to Chapter 4. Chapter 2 discusses the transform relationships from which we start, and Chapter 3 discusses the inversion technique to be used.

1.2. Methodology

The operating characteristic studied in this disseration is time-dependent server load. The server load at epoch t, denoted $W(t)$, is equal to the sum of the service times of customers waiting in the queue plus any remaining service time of a customer being served. The quantity $W(t)$ is also called virtual waiting time, because it is the time that a hypothetical customer arriving at epoch t would have to wait before beginning service. Our objective is to tabulate the expected value of server laod, $E[W(t)]$, for various epochs t and various system parameters (initial load, arrival rate, and service time distribution).

Our study of M/G/1 queueing systems can be embedded in the following general framework. Let $\{X(t), t \geq 0\}$ be a stochastic process

with stationary independent increments (a Lévy process) whose sample paths have no negative jumps. Let $W(t)$ be this same process modified by a reflecting barrier at zero. If $X(t) = S(t) - t$, where $\{S(t), t \geq 0\}$ is a compound Poisson process with positive jumps, then $W(t)$ is the server load process for an M/G/1 queue. (The jumps of $S(t)$ occur at customer arrival epochs and the jump sizes are service times.) We shall discuss two other choices for the X process which lead to W processes that provide bounds or approximations for the M/G/1 server load process. In one instance, we take $X(t)$ to be Brownian motion, and, in the other, we take $X(t) = G(t) - t$, where $G(t)$ is a gamma process (a Lévy process whose increments are gamma distributed).

In Chapter 2 we present a Laplace transform result for $E[W(t)]$ that covers both these two cases and the queueing (compound Poisson) case. In terms of its application to queueing processes, this result is valid for a system that begins operation either with or without an initial backlog of work and that has an average arrival rate less than, equal to, or greater than its average service rate.

There is no general closed-form expression for the exact inverse of the Laplace transform of mean server load, but numerical methods can be used to obtain an approximation of $E[W(t)]$ at a specific epoch t. The numerical technique employed in this research computes $E[W(t)]$ by taking a linear combination of the Laplace transform function evaluated at appropriate values of its argument. Theoretically, a more accurate approximation of $E[W(t)]$ can be obtained by using more evaluations of

the Laplace transform expression, but, when using an electronic computer for the computations, its finite word length places a limit on the accuracy that can be achieved in the $E[W(t)]$ approximation. Investigation of the technique using Laplace transforms with known inverses indicates that five or six significant digits for $E[W(t)]$ can be obtained efficiently; this is more than enough to justify confidence in the three or four digit results shown in the final $E[W(t)]$ tables.

The computational procedures used in this research could be applied to any M/G/1 queueing system whose service time distribution has a Laplace transform that can be evaluated numerically. In this research we achieve a balance between generality of results and ease of computation by concentrating on the well known class of M/G/1 queues whose service times have Erlang (or, equivalently, gamma) distributions. We denote these queueing systems $M/E_k/1$ and use the Erlang shape parameter k to describe the service times. (Parameter k is usually restricted to positive integer values when describing Erlang distributions, but here we allow k to assume any positive real value.) For example, the special case of $k = 1$ corresponds to the exponential service time distribution (an M/M/1 system). The present research also examines $M/E_k/1$ queues with k less than 1 and k greater than 1, corresponding to service time distributions with greater variance than the exponential and less variance, respectively. Most service time distributions encountered in actual practice can be adequately approximated using the very flexible Erlang family.

1.3. Overview of the Subsequent Chapters

In Chapter 2 we explain the sample path relationship between the server load process $W(t)$ and the associated net input process $X(t)$ in an M/G/1 queueing system. In this discussion, the net input process can actually be any Lévy process which has no negative jumps. Breiman [1968] gives an introduction to the theory of such processes, and Blumenthal and Getoor [1968] provide a comprehensive treatment. We then present a result developed by Harrison [1977] for the Laplace transform of $E[W(t)]$ when the net input is a general Lévy process. This theoretically oriented chapter finally examines a scaling procedure which facilitates comparisons among the various processes.

Chapter 3 explains the Laplace inversion technique developed by Gaver [1966] which gives approximation based on the expected value of an observational density function. The accuracy of Gaver's algorithm was improved by Stehfest [1970], and we test the technique using Laplace transform functions whose exact inverses are known. These numerical results are also compared with results from Veillon [1974] which were obtained using other Laplace inversion techniques. We then apply the technique to $E[W(t)]$ in $M/E_k/1$ queueing systems, and we compare our results with Coleman [1975] who obtained exact $E[W(t)]$ for the M/M/1 case by evaluating sums of Bessel functions. After some additional checks to ensure the accuracy of our approximations, we provide documentation of the computer programs which generate the tables of $E[W(t)]$. Gaver [1966, 1968] presented limited numerical results for transient

$E[W(t)]$ in several M/G/1 queues, thereby demonstrating the potential usefulness of his technique, and Coleman [1975] presented exact results only for the M/M/1 case. The major contribution of the present research is the extensive set of tables in Appendix C, providing transient mean server load in queues with a wide range of traffic intensities, initial loads, and service time distributions.

Chapter 4 explains how the scaled results in the tables can be used by a practitioner to determine $E[W(t)]$ in M/G/1 queues. Charts of the Erlang service time distributions are presented, and simple methods of interpolation in the tables are explained. Several sample problems demonstrate the entire procedure. It is intended that Chapter 4 can be used by a practitioner without recourse to Chapters 2 or 3.

The dissertation concludes in Chapter 5 with a brief summary, some qualitative observations, and suggestions for further research.

CHAPTER 2

THEORETICAL RESULTS FOR THE SERVER LOAD PROCESS

2.1. The Server Load Process in M/G/1 Queues

The M/G/1 queueing system consists of a group of customers, a waiting room, and a service facility. We assume that customer arrivals are described by a stationary Poisson process $\{A(t); t \geq 0\}$ with mean arrival rate λ, where $A(t)$ is the number of customers arriving during the time interval $[0,t]$. Thus, the probability of n arrivals in $[0,t]$ is

$$P\{A(t) = n\} = \frac{e^{-\lambda t}(\lambda t)^n}{n!} , \qquad n = 0, 1, \ldots ,$$

and the times between consecutive arrivals are exponentially distributed with mean $1/\lambda$. We further assume that there are no bulk arrivals, no reneging, and no balking. The number of potential customers and the size of the waiting room are assumed to be unlimited, and the queue discipline is first-come-first-served.

Customers arrive at epochs $\{t_1, t_2, \ldots\}$, each with a demand for service $\{S_1, S_2, \ldots\}$. These individual customer service times are independent of the interarrival times and are independent, identically distributed non-negative random variables with finite mean $E(S)$ and finite second moment $E(S^2)$. We assume that random variable S has a distribution function $F(\cdot)$ with corresponding Laplace-Stieltjes Transform (LST):

$$F^*(s) = E(e^{-sS}) = \int_0^\infty e^{-sx}\, dF(x) , \qquad 0 < s < \infty .$$

The general approach used in this research requires only that this LST can be evaluated numerically. For the specific cases to be investigated, S has a continuous distribution with density function $f(\cdot)$, so the LST reduces to the ordinary Riemann integral

$$\int_0^\infty e^{-sx} f(x)\, dx .$$

In particular, we examine M/G/1 systems with gamma distributed service times, but we employ the terminology usually reserved for the Erlang family of distributions, where the two parameters are the mean $E(S)$ and the shape parameter k. The Erlang density function is

$$(2.1.1) \qquad f(x) = \frac{[k/E(S)]^k}{(k-1)!} x^{k-1} e^{-kx/E(S)} , \qquad x \geq 0 ,$$

with $k \geq 1$ restricted to integer values. However, here we allow k to assume any non-negative real value, requiring that the factorial $(k-1)!$ in the density function be replaced by the gamma function,

$$\Gamma(k) = \int_0^\infty x^{k-1} e^{-x}\, dx .$$

Using the Erlang terminology, the LST of our service time distribution is (see Drake [1967, p. 138] for a derivation)

(2.1.2) $$F^*(s) = \left[\frac{k}{k + sE(S)}\right]^k , \qquad k \geq 0 ,$$

The variance of an Erlang random variable is $[E(S)]^2/k$. Thus, for various Erlang service time distributions with the same mean, the shape parameter k is inversely proportional to the variability of those service times. For example, Erlang service times with $0 < k < 1$ have even more variability than an exponential distribution $(k = 1)$. On the other hand, for very large values of k the variance is very small, and we approach the case of a constant or deterministic service time (an M/D/1 system) as k tends to infinity. Probability density functions for $0 < k \leq 1$ and $k \geq 1$ are shown graphically in Figure 4.1. Hillier and Lieberman [1974, p. 417] state that "empirical service-time distributions can usually be reasonably approximated by an Erlang distribution."

The most important descriptive parameter for a queueing system is its traffic intensity ρ, defined as the mean service time divided by the mean interarrival time. Thus, for M/G/1 systems, we have

$$\rho = \lambda E(S) \quad .$$

When ρ is less than one, i.e., when the average time between arrivals is greater than the average service time, we say that the system is "stable". In this case, ρ is the fraction of time that the server is busy, and it can be interpreted as the system's "utilization factor". Furthermore, for $\rho < 1$ the distribution of virtual waiting time

approaches an equilibrium distribution as t increases. Let W denote this steady-state waiting time. Its expected value is given by the Pollaczek-Khintchine formula,

$$E(W) = \frac{\lambda E(S^2)}{2(1-\rho)} , \qquad \rho < 1 . \tag{2.1.3}$$

Our numerical results for time-dependent virtual waiting time will allow us to observe how quickly this steady-state condition is approached. However, we will also examine "unstable" systems ($\rho \geq 1$), where the queue length and waiting time tend to increase without bound, so that no steady-state conditions exist.

As mentioned in the introductory chapter, the virtual waiting time or server load, $W(t)$, represents the work backlog at epoch t, i.e., the accumulated, unserviced demand. We define $W(t)$ in terms of the underlying work input processes, which allows us to use existing Laplace transform results for reflected Lévy processes. Sample path relationships are shown graphically in Figures 2.1 and 2.2, illustrating one possible realization for these stochastic processes.

The input process $S(t)$ represents the amount of customer work that arrives during the interval $[0,t]$ and is defined by

$$S(t) = S_1 + \cdots + S_{A(t)} , \qquad t \geq 0 .$$

This compound Poisson process has mean $E[S(t)] = \rho t$ and variance $Var[S(t)] = \lambda E(S^2)t$.

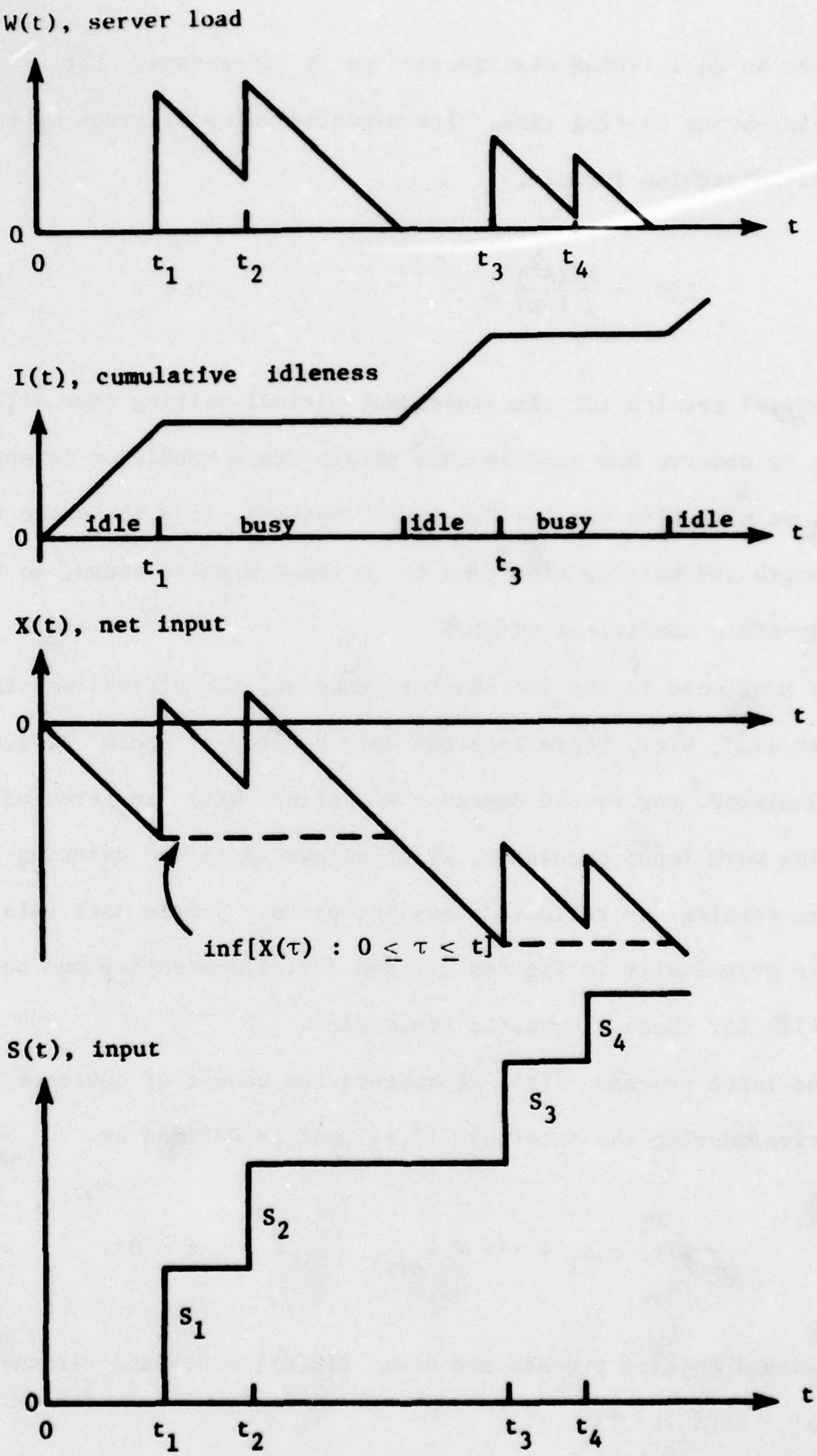


FIGURE 2.1. Sample Path Relationships when $W(0) = 0$.

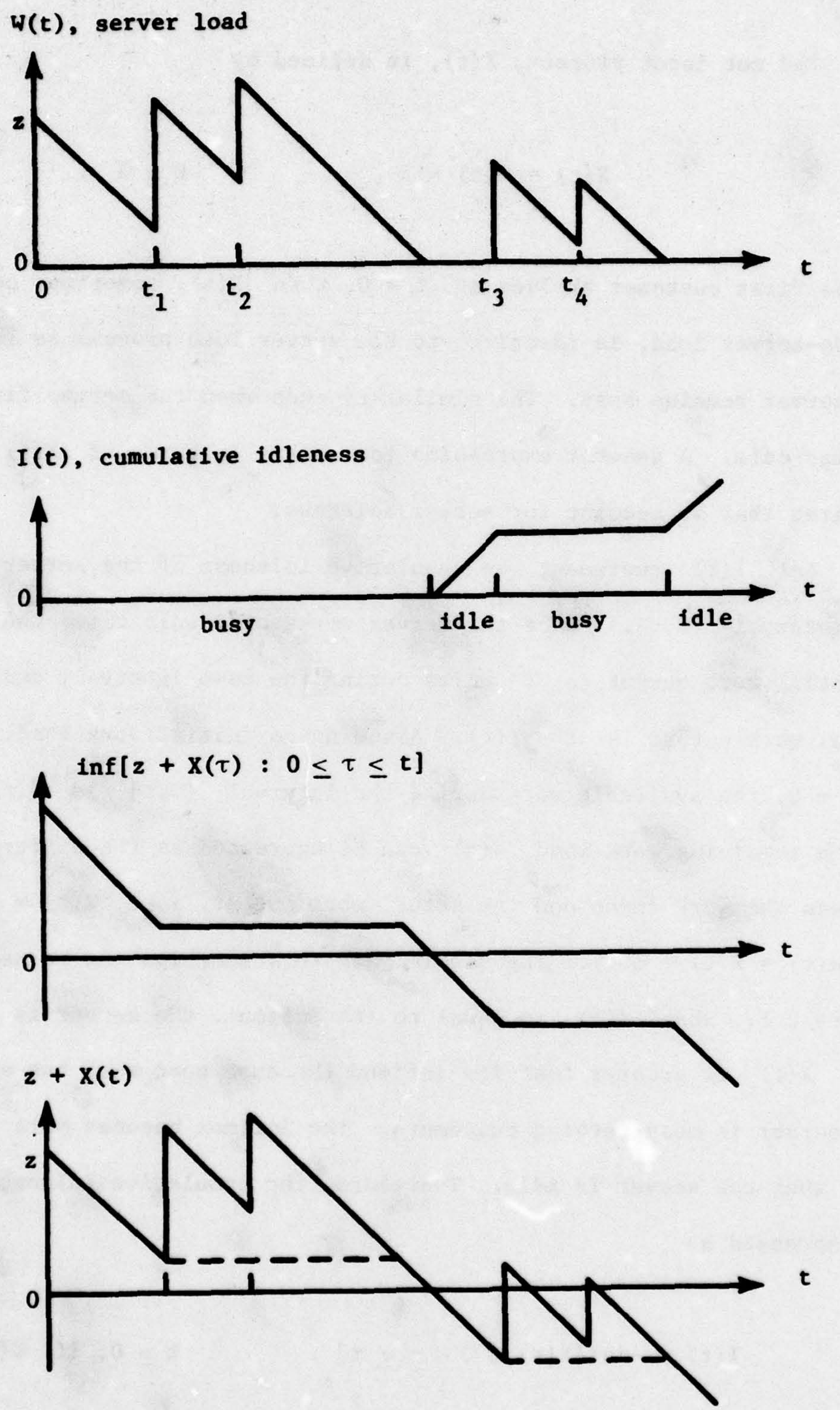


FIGURE 2.2. Sample Path Relationships when $W(0) = z > 0$.

The net input process, $X(t)$, is defined by

$$X(t) = S(t) - t \, , \qquad t \geq 0 \, .$$

If the first customer arrives at $t = 0$, then $X(t)$, sometimes called pseudo-server load, is identical to the server load process as long as the server remains busy. The similarity ends when the server first becomes idle. A general expression for $W(t)$ in terms of $X(t)$ requires that we account for server idleness.

Let $I(t)$ represent the cumulative idleness of the server during the interval $[0,t]$. Since the server works at a unit rate, the potential work output is t units during the same interval, and the actual work output is $t - I(t)$. Assuming no initial work load, i.e., $W(0) = 0$, the available work during the interval $[0,t]$ is $S(t)$, so the remaining work load $W(t)$ can be expressed as the difference between the work input and the actual work output, i.e., $W(t) = S(t)-[t-I(t)]$, or $W(t) = X(t) + I(t)$. The sample path relationships can be seen in Figure 2.1. When $X(t)$ is equal to its infimum, the server is idle. When $X(t)$ is greater than its infimum (because some work has arrived), the server is busy serving customers. The infimum becomes more negative only when the server is idle. Therefore, the cumulative idleness can be expressed as

$$I(t) = -\inf[X(\tau) : 0 \leq \tau \leq t] \, , \qquad t \geq 0, \text{ if } W(0) = 0 \, .$$

Combining this with the net input, we have the server load representation (no initial workload),

$$W(t) = X(t) - \inf[X(\tau) : 0 \leq \tau \leq t] , \qquad t \geq 0, \text{ if } W(0) = 0 .$$

The same reasoning applies to the more general case of initial server load, illustrated in Figure 2.2, where the underlying $S(t)$ and $X(t)$ sample paths would be identical to those shown in Figure 2.1. The initial server load z represents a specific amount of work available for service at epoch $t = 0$. Then the available work during an interval $[0,t]$ is $z + S(t)$, and the remaining work load $W(t)$ at any epoch t is $z + S(t) - [t - I(t)]$, or $W(t) = z + X(t) + I(t)$. The cumulative idleness function $I(t)$ is slightly more complicated when $W(0) > 0$. The server is initially busy in this case, so when $z + X(t)$ is equal to its infimum, the server is idle only if that infimum is negative. After the initial busy period, the infimum of $z + X(t)$ becomes more negative only when the server is idle, and the representation of the cumulative idleness function is similar to the case with no initial load. Thus, for this general case the server load representation is

$$W(t) = z + X(t) - \{\inf[z + X(\tau) : 0 \leq \tau \leq t]\}^{-} ,$$
$$t \geq 0, \text{ if } W(0) = z \geq 0 .$$

Our main objective is to compute the expected value of $W(t)$, where the expectation is taken with respect to a probability distribution over all possible sample paths. We adopt the notation $E_z[W(t)]$ and $E_z[I(t)]$ for expected server load and expected cumulative idleness, respectively, conditional on initial work load $W(0) = z$. From our discussion of the sample path relationships it follows that

$$E_z[W(t)] = E_z[z + S(t) - t + I(t)] \ ,$$

(2.1.3)

$$E_z[W(t)] = z + \rho t - t + E_z[I(t)] \ .$$

Thus, in M/G/1 queueing systems the mean server load can be separated into four additive terms: initial work load, new work input, potential work output, and cumulative idleness. Another useful observation is that $E_z[I(t)]$ must be zero in the interval from $t = 0$ to $t = z$, i.e., it is impossible for the server to become idle before the initial work load has been serviced. In Chapter 3 this property is used to provide a check on the accuracy of our numerical results.

Thus far the sample path relationships and expectations have been discussed in the context of M/G/1 queueing systems. However, we also calculate $E_z[W(t)]$ for processes where the same relationships apply, but where $S(t)$ is not compound Poisson. These related processes provide bounds or approximations for M/G/1 mean server load.

2.2. The Laplace Transform Result for Expected Server Load

All of the stochastic processes for which numerical values are calculated here can be discussed simultaneously in the following unified framework. Let $X = \{X(t), t \geq 0\}$ be a process with stationary, independent increments (an infinitely divisible or Lévy process) and no negative jumps, and define

$$-E[X(t)] = \mu t, \qquad \text{where} \quad -\infty < \mu < \infty ,$$

(2.2.1)

$$\mathrm{Var}[X(t)] = \sigma^2 t, \qquad \text{where} \quad 0 < \sigma^2 < \infty ,$$

for $t \geq 0$. According to the standard results for Lévy processes, the Laplace transform of $X(t)$ has the form

$$E[e^{-sX(t)}] = e^{-\Phi(s)t} \qquad \text{for } s > 0 \text{ and } t \geq 0 ,$$

and the exponent function $\Phi(\cdot)$ is convex with

$$\Phi(0) = 0, \quad \Phi'(0) = \mu , \quad \text{and} \quad \Phi''(0) = \sigma^2 , \tag{2.2.2}$$

cf. Harrison [1977] and Takacs [1967]. It can be shown that for each $s > 0$ there exists a unique $\omega(s) > 0$ such that

$$\Phi[\omega(s)] = s . \tag{2.2.3}$$

In the previous section we discussed the sample path relationships where process W is obtained from X by imposing a reflecting barrier at zero. Using the notation $E_z[W(t)] = E[W(t) | W(0) = z]$, let $P_W(z,s)$ denote the Laplace transform of $E_z[W(t)]$, that is,

$$P_W(z,s) = \int_0^\infty e^{-st} E_z[W(t)]\, dt \;, \qquad s > 0 \;.$$

Harrison [1977] developed a simple formula for the Laplace transform of $E_z[W(t)]$, where μ is unrestricted in sign:

$$(2.2.4) \qquad P_W(z,s) = \frac{z}{s} - \frac{\mu}{s^2} + \frac{e^{-\omega(s)z}}{s\omega(s)} \quad .$$

In Chapter 3 we use this formula to obtain accurate approximations of $E_z[W(t)]$ for specific epochs t, initial loads z, and various net input processes X. To achieve the desired accuracy the numerical inversion technique requires that $P_W(z,s)$ be computed for 34 values of s in order to obtain $E_z[W(t)]$ at a single epoch, and each evaluation of $P_W(z,s)$ requires that the functional equation (2.2.3) be solved to obtain $\omega(s)$.

The ease with which $\omega(s)$ can be calculated depends upon the form of the exponent function, and the exact form of $\Phi(\cdot)$ depends upon the process X. If we specify $X(t) = S(t) - t$, where $S(t)$ is a compound Poisson process, then it can be shown that

$$\Phi(s) = s - \lambda[1 - F^*(s)] \quad ,$$

In the previous section we discussed the sample path relationships where process W is obtained from X by imposing a reflecting barrier at zero. Using the notation $E_z[W(t)] = E[W(t)|W(0) = z]$, let $P_W(z,s)$ denote the Laplace transform of $E_z[W(t)]$, that is,

$$P_W(z,s) = \int_0^{\infty} e^{-st} E_z[W(t)]\, dt , \qquad s > 0 .$$

Harrison [1977] developed a simple formula for the Laplace transform of $E_z[W(t)]$, where μ is unrestricted in sign:

$$(2.2.4) \qquad P_W(z,s) = \frac{z}{s} - \frac{\mu}{s^2} + \frac{e^{-\omega(s)z}}{s\omega(s)} .$$

In Chapter 3 we use this formula to obtain accurate approximations of $E_z[W(t)]$ for specific epochs t, initial loads z, and various net input processes X. To achieve the desired accuracy the numerical inversion technique requires that $P_W(z,s)$ be computed for 34 values of s in order to obtain $E_z[W(t)]$ at a single epoch, and each evaluation of $P_W(z,s)$ requires that the functional equation (2.2.3) be solved to obtain $\omega(s)$.

The ease with which $\omega(s)$ can be calculated depends upon the form of the exponent function, and the exact form of $\Phi(\cdot)$ depends upon the process X. If we specify $X(t) = S(t) - t$, where $S(t)$ is a compound Poisson process, then it can be shown that

$$\Phi(s) = s - \lambda[1 - F^*(s)] ,$$

cf. Prabhu [1965, p. 70] and Takacs [1967, p. 59]. Using the terminology of M/G/1 queueing theory λ is the average arrival rate and $F^*(\cdot)$ is the Laplace transform of the service time distribution. Since $S(t)$ has mean $\lambda E(S)t$ and variance $\lambda E(S^2)t$, it follows that the parameters of X defined by equations (2.2.1) are $\mu = 1-\rho$ and $\sigma^2 = \lambda E(S^2)$. If $F^*(\cdot)$ can be evaluated numerically, then the properties (2.2.2) of $\Phi(\cdot)$ indicate that the functional equation $\Phi[\omega(s)] = s$ can be solved efficiently using an elementary one-dimensional search technique. In Chapter 3 we discuss our use of the Newton-Raphson method to obtain $\omega(s)$ for $M/E_k/1$ queueing systems.

For the special case of an M/M/1 system the LST of the service time distribution is obtained by setting $k = 1$ in equation (2.1.2), i.e., $F^*(s) = 1/[1 + sE(S)]$. For a specified value of s the functional equation $\Phi[\omega(s)] = s$ is a quadratic function of $\omega(s)$, and the positive solution is

$$(2.2.5) \qquad \omega(s) = \frac{-\{1 - \lambda[s+E(S)]\} + \sqrt{\{1 - \lambda[s+E(S)]\}^2 + 4sE(S)}}{2E(S)} .$$

The M/M/1 system is the only queue studied here that has an analytic solution to (2.2.3). In general the M/D/1 and $M/E_k/1$ cases will require a search to determine $\omega(s)$. The formulas for these cases are summarized in Table 2.3 below.

Two other choices for $X(t)$ yield reflected processes $W(t)$ which provide bounds or approximations for M/G/1 server load. The first case is the Wiener process, which has been used as a model for a variety of physical processes since its original development as a

	Gamma Input Process	M/G/1 Queueing Systems			Brownian Motion Process
		$M/E_k/1$	M/M/1	M/D/1	
Input Process	$G(t)$ with parameters α,β	$S(t) = S_1 + \cdots + S_{A(t)}$ where $A(t)$ is Poisson			—
Characteristics of the Input Process	gamma distributed increments	S_i Erlang with shape parameter k	S_i exponential $(k = 1)$	S_i constant $(k \to \infty)$	—
Net Input Process $X(t)$	$X_G(t) = G(t)-t$	$X(t) = S(t) - t$			$X_W(t) = \sigma\xi(t)-\mu t$ (ξ is standard Wiener process)
$E[X(t)]$	$(\frac{\alpha}{\beta} - 1)t$	$(\lambda E(S) - 1)t$	$(\lambda E(S) - 1)t$	$(\lambda E(S) - 1)t$	$-\mu t$
$Var[X(t)]$	$(\frac{\alpha}{\beta^2} - 1)t$	$\lambda \frac{k+1}{k} E(S)^2 t$	$\lambda 2E(S)^2 t$	$\lambda E(S)^2 t$	$\sigma^2 t$
Exponent Function $\Phi(s)$	$s - \alpha \ln(1+\frac{s}{\beta})$	$s - \lambda\{1 - [\frac{k}{k+sE(S)}]^k\}$	$s - \frac{sE(S)}{1 + sE(S)}$	$s - \lambda[1 - e^{-sE(S)}]$	$\mu s + \frac{1}{2} \sigma^2 s^2$
Solution technique for $\Phi[\omega(s)] = s$	Newton-Raphson Search	Newton-Raphson Search	Quadratic Formula	Newton-Raphson Search	Quadratic Formula

TABLE 2.3. Summary of Characteristics for the Processes under Study.

model for Brownian motion. Gaver [1968] proposed that this diffusion process could be used as an approximation for the net input process of an M/G/1 queue by equating the first and second moments of the two processes.

The Brownian motion process is denoted by

$$X_B(t) = \sigma\xi(t) - \mu t \quad ,$$

where $\xi(t)$ is a Wiener rocess whose stationary, independent increments have a normal distribution with mean zero and unit variance. It follows that $X_B(t)$ has mean $-\mu t$ and variance $\sigma^2 t$. As previously noted, the net input process of an M/G/1 queue has mean $E[X(t)] = (\rho-1)t$ and variance $\mathrm{Var}[X(t)] = \lambda E(S^2)t$. Thus, using $X_B(t)$ to approximate $X(t)$ we would take $-\mu = \rho-1$ and $\sigma^2 = \lambda E(S^2)$. The exponent function for Brownian motion is $\Phi(s) = \mu s + \frac{1}{2}\sigma^2 s^2$, cf. Takacs [1967, p. 81]; thus the solution of $\Phi[\omega(s)] = s$ can be calculated using the quadratic formula.

The second non-queueing input process that we consider as an approximation is a gamma input process, denoted

$$X_G(t) = G(t) - t \quad ,$$

where $G(t)$ is a process whose stationary, independent increments have a gamma distribution. The gamma density function is

$$f(x) = \frac{\beta}{\Gamma(\alpha)} (\beta x)^{\alpha-1} e^{-\beta x} \,, \qquad \alpha > 0,\ \beta > 0,\ x \geq 0 \quad ,$$

so $G(t)$ has mean $(\alpha/\beta)t$ and variance $(\alpha/\beta^2)t$. It follows directly that $X_G(t)$ has mean $(\alpha/\beta - 1)t$ and variance $(\alpha/\beta^2)t$. Thus, using $X_G(t)$ to approximate the net input process of an M/G/1 queue we would choose α and β such that $\alpha/\beta - 1 = \rho - 1$ and $\alpha/\beta^2 = \lambda E(S^2)$. The exponent function for the gamma input process is $\Phi(s) = s - \alpha \log(1 + s/\beta)$, cf. Takacs [1967, p. 66]; the solution of the functional equation (2.2.3) requires a search technique.

2.3. Scaling the Reflected Levy Processes

In this section we present a method for normalizing the processes of interest by using a simple linear tranformation for both the epochs and the server load. This technique allows us to reduce the tabulations required to describe actual operating systems and also facilitates comparisons among different processes by adopting a common, normalized time scale.

Consider an M/G/1 queueing system (as originally described in Section 2.1) with service times $S_1, S_2, \ldots$ having distribution function $F(\cdot)$ with mean $E(S)$ and second moment $E(S^2)$. Let $\{A(t);\ t \geq 0\}$ be the Poisson arrival process with mean arrival rate λ. The net input process is

$$X(t) = S_1 + \cdots + S_{A(t)} - t \quad ,$$

and the server load process $W(t)$ is obtained from $[z + X(t)]$ by the reflection mapping described in Section 2.1, where z is the initial server load $W(0)$.

We define a new scaled process in terms of the original queueing process

$$X^*(t) = aX(bt) \quad .$$

This scaled net input process has the form

$$(2.3.1) \qquad X^*(t) = S_1^* + \cdots + S_{A^*(t)}^* - ct \ , \qquad t \geq 0 \ ,$$

where $c = ab$, $A^*(t) = A(bt)$, and $S_i^* = aS_i$. Note that the random variables S_i^* have distribution function $G^*(x) = F(x/a)$, and that $A^*(t)$ is a Poisson process with mean arrival rate $\lambda^* = b\lambda$. Let μ^* and σ_*^2 be defined by

$$(2.3.2) \qquad \begin{aligned} E[X^*(t)] &= -\mu^* t \ , \\ \mathrm{Var}[X^*(t)] &= \sigma_*^2 t \ . \end{aligned}$$

We accomplish our normalization by choosing a and b so that $|\mu^*| = 1$ and $\sigma_*^2 = 1$.

By substituting $aX(bt)$ for $X^*(t)$ on the left hand side of equations (2.3.2) and then using the parameters of process X as

defined by equations (2.2.1), it is easy to show that this particular scaling requires that

$$a = \frac{|\mu|}{\sigma^2} \quad \text{and} \quad b = \frac{\sigma^2}{\mu^2} .$$

If we define W^* as the reflection of $[z^* + X^*(t)]$, where the scaled initial server load is $z^* = az$, then it is easy to verify that $W^*(t) = aW(bt)$. That is, the reflection of the scaled net input process is identical to the scaled version of the original server load process. If we let t^* represent a (scaled) epoch for the scaled process, then it follows that

$$E_{z^*}[W^*(t^*)] = \frac{|\mu|}{\sigma^2} E_z[W(\frac{\sigma^2}{\mu^2} t^*)] , \qquad z = \frac{\sigma^2}{|\mu|} z^* .$$

Thus, we can obtain scaled mean server load by evaluating an original queueing system and applying the transformations.

Before discussing the scaled process further, consider a second queueing system having Poisson arrival process $A'(t)$ with mean rate $\lambda' \neq \lambda$ and service times $S_1', S_2', \ldots$ with distribution function $F'(x) = F(\lambda' x/\lambda)$ so that $E(S') = \lambda E(S)/\lambda'$. The traffic intensity parameter for this second queueing system is the same as the original, that is,

$$\rho' = \lambda' E(S') = \lambda' \cdot \lambda E(S)/\lambda' = \lambda E(S) = \rho ,$$

defined by equations (2.2.1), it is easy to show that this particular scaling requires that

$$a = \frac{|\mu|}{\sigma^2} \quad \text{and} \quad b = \frac{\sigma^2}{\mu^2} .$$

If we define W^* as the reflection of $[z^* + X^*(t)]$, where the scaled initial server load is $z^* = az$, then it is easy to verify that $W^*(t) = aW(bt)$. That is, the reflection of the scaled net input process is identical to the scaled version of the original server load process. If we let t^* represent a (scaled) epoch for the scaled process, then it follows that

$$E_{z^*}[W^*(t^*)] = \frac{|\mu|}{\sigma^2} E_z[W(\frac{\sigma^2}{\mu^2} t^*)] , \qquad z = \frac{\sigma^2}{|\mu|} z^* .$$

Thus, we can obtain scaled mean server load by evaluating an original queueing system and applying the transformations.

Before discussing the scaled process further, consider a second queueing system having Poisson arrival process $A'(t)$ with mean rate $\lambda' \neq \lambda$ and service times $S_1', S_2', \ldots$ with distribution function $F'(x) = F(\lambda' x/\lambda)$ so that $E(S') = \lambda E(S)/\lambda'$. The traffic intensity parameter for this second queueing system is the same as the original, that is,

$$\rho' = \lambda' E(S') = \lambda' \cdot \lambda E(S)/\lambda' = \lambda E(S) = \rho ,$$

but the variance parameter is different:

$$\sigma'^2 = \lambda' \; E(S'^2) = \lambda'(\frac{\lambda}{\lambda'})^2 \; E(S^2) = \frac{\lambda}{\lambda'} \lambda E(S^2) = \frac{\lambda}{\lambda'} \sigma^2 \; .$$

This second system and the original are related by

$$X'(t) = \frac{\lambda}{\lambda'} X(\frac{\lambda'}{\lambda} t) \quad ,$$

or equivalently by

$$X'(t) = \frac{E(S')}{E(S)} X(\frac{E(S)}{E(S')} t) \quad .$$

In the context of $M/E_k/1$ queueing systems, the second queueing system has the same ρ and same Erlang shape parameter k as the original, but the different values for σ^2 and σ'^2 require simple transformation of the epoch scale and waiting time scale.

Returning to the scaled process which was defined in terms of the original queueing process, we now express the original process in terms of the second process:

$$X^*(t) = \frac{|\mu|}{\sigma^2} X(\frac{\sigma^2}{\mu^2} t) = \frac{|\mu|}{\sigma^2} \frac{\lambda}{\lambda'} X'(\frac{\lambda'}{\lambda} \frac{\sigma^2}{\mu^2} t) = \frac{|\mu|}{\sigma'^2} X'(\frac{\sigma'^2}{\mu^2} t) \quad .$$

Once ρ (or μ) has been specified and a specific form of the service time distribution (i.e., a specific Erlang shape parameter k) has been determined, then we are free to choose the variance parameter (i.e.,

the time scales) for the queueing process from which X^* will be obtained.

This research tabulates $E_{z^*}[W^*(t^*)]$ for various values of ρ, k, and z^*, and one can obtain mean server load for an actual queueing system by selecting the table with the closest (ρ, k, z^*) combination or by interpolating between two tabulated scaled processes, and then applying the transformation:

$$E_z[W(t)] = \frac{\sigma^2}{|\mu|} E_{z^*}[W^*(\frac{\mu^2}{\sigma^2} t)] , \qquad z^* = \frac{|\mu|}{\sigma^2} z .$$

Since $\mu = 1-\rho$, this particular normalization cannot be used for systems with $\rho = 1$, i.e., $\mu = 0$. In this case we tabulate mean server load for Erlang queueing systems with λ and $E(S)$ chosen so that the variance parameter $\sigma^2 = \lambda E(S^2) = 1$. Let superscript zero indicate parameters for the tabulated system. Then $\sigma^2 = 1$ requires that

$$\lambda^0 = \frac{k+1}{k} \qquad \text{and} \qquad E(S^0) = \frac{k}{k+1} .$$

Mean server load for an actual operating system with shape parameter k and mean arrival rate λ can be obtained from the tabulated values as follows:

$$E_z[W(t)] = \frac{\lambda^0}{\lambda} E_{z^0}[W^0(\frac{\lambda}{\lambda^0} t)] = \frac{k+1}{k\lambda} E_{z^0}[W^0(\frac{k\lambda}{k+1} t)] , \quad z^0 = \frac{k\lambda}{k+1} z .$$

The tabulated values for the $\rho = 1$ case correspond to an actual queueing system, whereas the tabulated values for the $\rho \neq 1$ cases represent a stochastic process with "potential work output rate" of $c = ab = 1/|\mu|$, interpreted from equation (2.3.1). For the $\rho < 1$ cases, mean server load in queueing systems approaches the Pollaczek-Khintchine values, equation (2.1.3). Thus, the tabulated values for all scaled processes approach

$$\frac{|\mu|}{\sigma^2} \cdot \frac{\sigma^2}{2|\mu|} = \frac{1}{2} ,$$

thereby allowing comparisons concerning approach to equilibrium.

CHAPTER 3

NUMERICAL INVERSION OF THE LAPLACE TRANSFORM

3.1. Approximation by an Expected Value

In the previous chapter we cited an expression for the Laplace transform for the three processes of interest. For the two special cases of Brownian motion and the M/M/1 queue, analytic methods can be applied and exact numerical solutions can be obtained. However, in general it is necessary to use a method for numerical inversion of the Laplace transform in order to evaluate the process of interest. In this chapter we describe such a method and its implementation.

The first two sections of the chapter describe the method for approximate transform inversion using an expected value. In the third section we present numerical results for test cases where this method is applied to Laplace transforms whose exact inverses are known, and we describe the computer routines that apply the inversion method to prepare tables of $E[W(t)]$.

The method used in this research for numerical inversion of the Laplace transform was originally developed by Gaver [1966]. Although there are other techniques which could have been implemented, this particular method was chosen because it had been applied successfully to the Laplace transform expression for $E[W(t)]$ in the M/M/1 and M/G/1 queues by Gaver [1966, 1968].

The problem of inverting the Laplace transform can be summarized as follows. We have a function of interest, $P(t)$, for which no closed-form analytic expression is known, so that it cannot be evaluated

numerically. However, we do know the expression for its Laplace transform,

$$p(s) = \int_0^\infty e^{-st} P(t)\, dt \quad .$$

Specifically, in this research the function or process of interest is $E[W(t)]$ and the Laplace transform expression was cited in Chapter 2. In general, we wish to tabulate $P(t)$ for various values of t.

The method presented here computes an approximate value of the function at a specified point t'. Theoretically this approximation can be computed as precisely as desired. Consider observing the function of interest at a random time T that has density function $f(t)$ such that the values of T are concentrated near t', as shown in Figure 3.1. Then $\bar{P}(t')$, an approximation of $P(t')$, can be expressed as

$$\bar{P}(t') = \int_0^\infty P(t) f(t)\, dt \; . \tag{3.1.1}$$

Now the problem is one of finding an observational density function $f(t)$ so that the right hand side of (3.1.1) can be expressed in terms of $p(s)$.

Gaver [1966] examined such a family of density functions that are well-suited to numerical computation. Let $f_n(t; a)$ be the density function for random variable T, with parameters n and a, as follows:

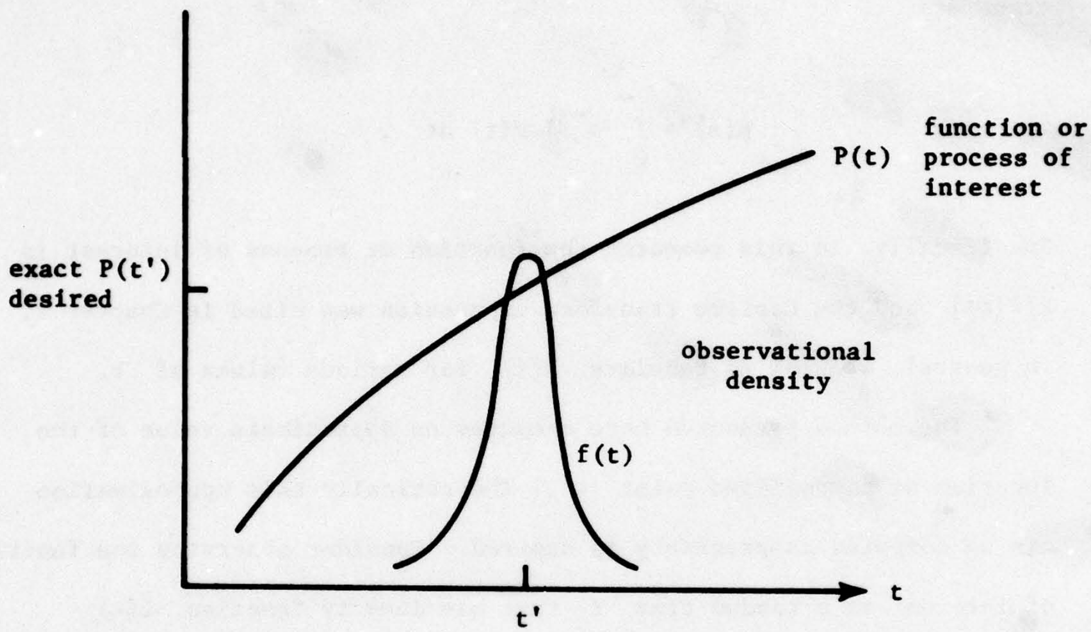


FIGURE 3.1. Observational Density Function

$$(3.1.2) \qquad f_n(t;\, a) = a\,\frac{(2n)!}{n!\,(n-1)!}\,(1 - e^{-at})^n\, e^{-nat}\ , \qquad a > 0,\ n = 1,2,\ldots\ .$$

Gaver [1966] showed that $f_n(t;\, a)$ has the following properties:

$$\text{modal value} \approx \frac{1}{a}\ln 2 \qquad ,$$

$$\text{Var}(T) \approx \frac{1}{a^2}\,\frac{2n+2}{(2n-1)\,(4n+1)}\ .$$

For increasing values of n, $f_n(t;\, a)$ becomes more sharply peaked at $t = 1/a \ln 2$. Using $f_n(t;\, a)$ as the observational density function

in (3.1.1), we let the approximation of $P(t')$ be

$$\bar{P}_n = \int_0^{\infty} P(t)\, f_n(t;\, a)\, dt \quad , \tag{3.1.3}$$

where parameter a is chosen such that $a = 1/t' \ln 2$, thereby concentrating the values of random variable T near t' as desired.

By substituting the observational density (3.1.2) in the approximating expression (3.1.3) and then expanding $(1 - e^{-at})^n$ by the binomial theorem, we obtain

$$\bar{P}_n = a \frac{(2n)!}{n!\,(n-1)!} \sum_{i=0}^{n} \binom{n}{i} (-1)^i \, p[(n+i)a] \quad . \tag{3.1.4}$$

Thus, $\bar{P}_n$, an approximation of $P(t')$, is based on a linear combination of the Laplace transform $p(s)$ evaluated at $n+1$ different values of s. Theoretically, the sequence $\{\bar{P}_n;\ n = 1, 2, 3, \ldots\}$ converges to $P(1/a \ln 2)$, i.e., to $P(t')$. For any finite n, we can calculate $\bar{P}_n$ using (3.1.4), and because $Var(T) \to 0$ as $n \to \infty$ we can obtain as good an approximation as desired by using a large enough value of n. Practically, $p(s)$, the Laplace transform of $P(t)$, must be evaluated at $n+1$ values of s in order to evaluate $\bar{P}_n$; in some applications much time may be required to make each evaluation of the Laplace transform expression and so the cost of obtaining the desired precision in $\bar{P}_n$ may be prohibitive. Also, for large n the factorial terms in (3.1.4) are integers with too many digits to be precisely evaluated on computers with finite word length, and operations involving these large numbers

can result in considerable rounding errors. From the standpoint of computational efficiency it is important to improve the accuracy of the approximate inverse without requiring evaluation of $\bar{P}_n$ for extremely large values of n.

3.2. Improving the Accuracy of the Approximation

Gaver [1966] showed that each $\bar{P}_n$ can be represented as an asymptotic expansion, i.e.,

$$\bar{P}_n \approx P(t') + \frac{\alpha_1}{n} + \frac{\alpha_2}{n^2} + \frac{\alpha_3}{n^3} + \cdots \tag{3.2.1}$$

where the error components α_i depend only on t' and not on n. An improved approximation of $P(t')$ can be obtained by taking a linear combination of a set of the $\bar{P}_n$ approximations, where the values of n are integer powers of 2. This method, extrapolation to the limit, ensures that some of the error terms in (3.2.1) are cancelled out, cf. Gaver [1966] and Henrici [1964].

Stehfest [1970] demonstrated an improved calculation method that uses a linear combination of an even number of the $\bar{P}_n$ approximations; this method cancels even more of the error terms than Gaver's method (extrapolation to the limit). Stehfest combined the coefficients for the linear combination of $\bar{P}_n$ with the coefficients of $p(s)$ in (3.1.4) so that Fa, the approximation of $P(t')$, could be expressed directly as a linear combination of $p(s)$:

$$\text{(3.2.2)} \qquad Fa = \frac{\ln 2}{t'} \sum_{i=1}^{N} V_i \, p\left(\frac{\ln 2}{t'} i\right) \quad .$$

Here N must be even and the combined coefficients V_i are

$$\text{(3.2.3)} \qquad V_i = (-1)^{(N/2)+i} \sum_{k=[\frac{i+1}{2}]}^{\mathrm{Min}(i,\frac{N}{2})} \frac{k^{N/2} \, (2k)!}{(\frac{N}{2} - k)! \; k! \; (k-1)! \; (i-k)! \; (2k-1)!} \quad .$$

For a theoretical comparison of the original $\bar{P}_n$, extrapolation to the limit, and Fa, consider the three methods where the number of values of $p(s)$ is 32 in each case. Thirty-two evaluations of the $p(s)$ could be used to compute $\bar{P}_{31}$, and from (3.2.1) the first error term would be $\alpha_{31}/31$. On the other hand the thirty-two values of $p(s)$ could be used to compute $\bar{P}_1$, $\bar{P}_2$, $\bar{P}_4$, $\bar{P}_8$, and $\bar{P}_{16}$, and it could be shown that extrapolation to the limit would yield an approximation of $P(t')$ where the first four error terms of equation (3.2.1) would cancel completely. The resulting approximation would be $P(t') + \alpha_5/2^{10} + \cdots$. Finally using Stehfest's method with $N = 32$, the first fifteen error terms in equation (3.2.1) would cancel, and it could be shown that the approximation would be $Fa = P(t') - \alpha_{16}/16! + \cdots$. This theoretical superiority of Stehfest's method was verified numerically in preliminary investigations using functions with known inverses.

3.3. Computer Implementation of the Technique

In this research, the algorithm LINV developed by Stehfest [1970] was translated from ALGOL to FORTRAN IV. Some changes were made in evaluating the V_i of equation (3.2.3) to take advantage of cancelling the factorial terms and thereby avoid rounding errors. Some preliminary numerical investigations used BASIC and single and double precision FORTRAN IV. The computer programs listed in Appendix B specify double precision for all non-integer variables, and the "AUTODBL" option available with the IBM FORTRAN H-Extended Compiler was used in all of the research reported here. With this extended precision there are approximately 36 significant decimal digits for the non-integer variables.

Stehfest [1970] published a table of results applying LINV to six transforms whose inverses are known, using 8 digit arithmetic and $n = 10$. Figure 3.2 shows the exact values of the six functions, the approximations from this research using $N = 34$ with 36 digit arithmetic, and the original Stehfest approximations using $N = 10$. We observe that there are at least six correct significant digits in our approximations (using $N = 34$) for each of the test functions.

Since the LINV approximation (Fa) is based upon the original $\bar{P}_n$, it should also be more accurate when more values of the Laplace transform p(s) are used, i.e., if N is very large. However, for large values of N the rounding errors in evaluating V_i can affect the accuracy of the results. This problem was investigated by applying LINV to the six test functions of Figure 3.2; approximations were obtained using

3.3. Computer Implementation of the Technique

In this research, the algorithm LINV developed by Stehfest [1970] was translated from ALGOL to FORTRAN IV. Some changes were made in evaluating the V_i of equation (3.2.3) to take advantage of cancelling the factorial terms and thereby avoid rounding errors. Some preliminary numerical investigations used BASIC and single and double precision FORTRAN IV. The computer programs listed in Appendix B specify double precision for all non-integer variables, and the "AUTODBL" option available with the IBM FORTRAN H-Extended Compiler was used in all of the research reported here. With this extended precision there are approximately 36 significant decimal digits for the non-integer variables.

Stehfest [1970] published a table of results applying LINV to six transforms whose inverses are known, using 8 digit arithmetic and $n = 10$. Figure 3.2 shows the exact values of the six functions, the approximations from this research using $N = 34$ with 36 digit arithmetic, and the original Stehfest approximations using $N = 10$. We observe that there are at least six correct significant digits in our approximations (using $N = 34$) for each of the test functions.

Since the LINV approximation (Fa) is based upon the original $\bar{P}_n$, it should also be more accurate when more values of the Laplace transform p(s) are used, i.e., if N is very large. However, for large values of N the rounding errors in evaluating V_i can affect the accuracy of the results. This problem was investigated by applying LINV to the six test functions of Figure 3.2; approximations were obtained using

FIGURE 3.2 INVERSION OF TEST FUNCTIONS

T	EXACT F(T)	APPROXIMATE F(T) USING STEHFEST'S METHOD N=34	N=10	EXACT F(T)	APPROXIMATE F(T) USING STEHFEST'S METHOD N=34	N=10
	F(T) = 1/SQRT(PI*T)			F(T) = -C-LN(T)		
1.0	0.5641895543	0.5641895835	0.56555	-0.5772156715	-0.5772156649	-0.57782
2.0	0.3989422597	0.3989422804	0.39912	-1.2703628521	-1.2703628455	-1.27084
3.0	0.3257349910	0.3257350079	0.32655	-1.6758279602	-1.6758279536	-1.67544
4.0	0.2820947771	0.2820947918	0.28278	-1.9635100327	-1.9635100260	-1.96392
5.0	0.2523132391	0.2523132522	0.25174	-2.1866535840	-2.1866535773	-2.18727
6.0	0.2303294210	0.2303294330	0.22989	-2.3689751408	-2.3689751341	-2.36870
7.0	0.2132436076	0.2132436186	0.21322	-2.5231258206	-2.5231258140	-2.52270
8.0	0.1994711299	0.1994711402	0.19956	-2.6566572132	-2.6566572066	-2.65740
9.0	0.1880631848	0.1880631945	0.18814	-2.7744402489	-2.7744402422	-2.77390
10.0	0.1784124024	0.1784124116	0.17796	-2.8798007645	-2.8798007579	-2.88091
	F(T) = (T*T*T)/6			F(T) = EXP(-T)		
1.0	0.1666666667	0.1666666667	0.16568	0.3678794412	0.3678794412	0.36798
2.0	1.3333333333	1.3333333333	1.32543	0.1353352832	0.1353352832	0.13557
3.0	4.5000000000	4.5000000000	4.47354	0.0497870684	0.0497870684	0.05043
4.0	10.6666666667	10.6666666667	10.60342	0.0183156389	0.0183156389	0.01849
5.0	20.8333333333	20.8333333333	20.70845	0.0067379470	0.0067379470	0.00640
6.0	36.0000000000	36.0000000000	35.78831	0.0024787522	0.0024787522	0.00195
7.0	57.1666666667	57.1666666667	56.82535	0.0009118820	0.0009118820	0.00036
8.0	85.3333333333	85.3333333333	84.82735	0.0003354626	0.0003354625	-0.00006
9.0	121.5000000000	121.5000000000	120.78473	0.0001234098	0.0001234097	-0.00047
10.0	166.6666666667	166.6666666667	165.66750	0.0000453999	0.0000453999	-0.00020
	F(T) = SIN(SQRT(2*T))			F(T) = 1-3*T+3*T*T/2-T*T*T/6		
1.0	0.9877659460	0.9877659972	0.98775	-0.6666666667	-0.6666666667	-0.66533
2.0	0.9092974268	0.9092974740	0.91001	-0.3333333333	-0.3333333333	-0.32531
3.0	0.6381576351	0.6381576682	0.63826	1.0000000000	1.0000000000	1.02575
4.0	0.3080717424	0.3080717583	0.30968	2.3333333333	2.3333333333	2.39533
5.0	-0.0206835315	-0.0206835326	-0.02119	2.6666666667	2.6666666667	2.78844
6.0	-0.3169471632	-0.3169471796	-0.31927	1.0000000000	1.0000000000	1.21092
7.0	-0.5646958988	-0.5646959281	-0.57254	-3.6666666667	-3.6666666667	-3.32956
8.0	-0.7568024953	-0.7568025346	-0.76869	-12.3333333333	-12.3333333333	-11.82953
9.0	-0.8916822545	-0.8916823007	-0.91049	-26.0000000000	-26.0000000000	-25.28394
10.0	-0.9712777990	-0.9712778493	-0.98949	-45.6666666667	-45.6666666667	-44.88512

N equal to 30, 32, 34, and 36. In each case the correct number of decimal places was recorded, rounding off both the exact value and the approximation when making the comparison. Figure 3.3 shows the average number of correct decimal places over the ten values of t evaluated in each case. We observe that the approximations using N = 34 are at least as good as those using N = 32 or N = 36 in all cases. Thus, in the remainder of this research we will use N = 34 in the LINV algorithm.[1]

Figure 3.4 compares the LINV algorithm with other inversion techniques. The top section of this table reproduces some results by Dubner and by Veillon (1974)[2]; Dubner's approximations are based on 500 values of p(s), while Veillon's method uses 64 values. We observe that in all cases the LINV algorithm using 34 values of p(s) is at least as accurate as their techniques. The bottom section of Figure 3.4 compares approximations of a second function; we observe that once again the LINV results are as accurate as the results obtained by other techniques.

Figures 3.2, 3.3 and 3.4 indicate that LINV can provide very accurate approximations for the test functions, but it is also important to investigate LINV's accuracy using the function of interest in this

---

[1] In preliminary investigations using FORTRAN IV double precision (approximately 17 digit arithmetic) without the AUTODBL option, the most accurate approximations were obtained using N = 18.

[2] Reprinting privileges for the Stehfest results (Comm. of the ACM, Vol. 13, No. 1, Copyright 1970) and the results published by Veillon (Comm. of the ACM, Vol. 17, No. 10, Copyright 1974) were granted by permission of the Association for Computing Machinery.

FIGURE 3.3. Average Number of Correct Rounded-off Decimal Places in LINV Approximations of Six Functions for Determination of Optimal Value of N

Function	N, the number of values of the Laplace transform p(s) used by LINV to estimate the function			
	N = 30	N = 32	N = 34	N = 36
$\frac{1}{\sqrt{\pi t}}$	6.9	6.9	6.9	6.9
$\frac{t^3}{6}$	9.7	10.5	12.7	11.9
$\sin \sqrt{2t}$	6.7	6.7	6.7	6.7
$- C - \ln t$	7.3	7.3	7.5	7.4
e^{-t}	9.7	10.1	10.5	10.0
$1 - 3t + \frac{3t^2}{2} + \frac{t^3}{6}$	9.7	10.5	11.9	10.6

FIGURE 3.4 COMPARISONS WITH OTHER TECHNIQUES

F(T) = 1/SQRT(T*PI)

T	EXACT F(T)	APPROXIMATE F(T) USING NUMERICAL INVERSION: STEHFEST'S METHOD N=34	STEHFEST'S METHOD N=10*	DUBNER*	VEILLON*
1	0.56418955	0.56418958	0.56555	0.73172	0.56419
2	0.39894226	0.39894228	0.39912	0.40035	0.39894
3	0.32573499	0.32573501	0.32655	0.26343	0.32573
4	0.28209478	0.28209479	0.28278	0.28286	0.28209
5	0.25231324	0.25231325	0.25174	0.29365	0.25231
6	0.23032942	0.23032943	0.22989	0.22901	0.23033
7	0.21324361	0.21324362	0.21322	0.18062	0.21324
8	0.19947113	0.19947114	0.19956	0.20112	0.19947
9	0.18806318	0.18806319	0.18814	0.21609	0.18806
10	0.17841240	0.17841241	0.17796	0.17650	0.17841

F(T) = EXP(-T/2)

T	EXACT F(T)	APPROXIMATE F(T) USING NUMERICAL INVERSION: STEHFEST N=34	BELLMAN*	VEILLON*
4.140186	0.1261740274277	0.1261740274279	0.120527	0.126174
2.501126	0.2863434986283	0.2863434986283	0.288195	0.286329
1.643438	0.4396751282721	0.4396751282719	0.439084	0.439675
1.085084	0.5812687987062	0.5812687987065	0.581308	0.581269
0.693147	0.7071068437334	0.7071068437334	0.707318	0.707107
0.412298	0.8137118145453	0.8137118145451	0.813401	0.813712
0.214821	0.8981569124175	0.8981569124171	0.898482	0.898158
0.085541	0.9581312512155	0.9581312512156	0.957847	0.958135
0.016048	0.9920081067225	0.9920081067225	0.992205	0.992015

*NOTE: DATA FOR METHODS MARKED BY AN ASTERISK WERE TAKEN FROM A.C.M ALGORITHM 486, 'NUMERICAL INVERSION OF LAPLACE TRANSFORM', BY FRANCOISE VEILLON, COMMUNICATIONS OF THE A.C.M., VOLUME 17, NUMBER 10, OCTOBER 1974.

research, E[W(t)]. Our earlier investigations using LINV (not included here) verified the results for M/M/1 queues reported by Gaver [1968], but it is more appropriate to compare our LINV results with mean server load obtained by some method other than Laplace transform inversion. Fortunately, Coleman [1975] evaluated $E[W(t)]$ in the M/M/1 queue using a sum of Bessel functions. In Figure 3.5 the first three columns show his results at five epochs and the LINV results using equations (2.2.4) and (2.2.5) for the Laplace transform expression. The five significiant digits available in the Coleman results are matched exactly when the LINV results are rounded off.

As mentioned in Section 2.2, except for the Wiener process and the M/M/1 queue, the value of $\omega(s)$ must be obtained using a search technique. Our final results use the quadratic formula to determine $\omega(s)$ for the Wiener process, but in all other cases (M/D/1, $M/E_k/1$ including M/M/1, and the gamma input process) we use the standard Newton-Raphson method. This search technique was chosen because expressions for the derivatives of the functions were available and subsequent computations demonstrated that the search usually converged in only four or five iterations.

The exponent function $\Phi(\cdot)$ is designated FMD1, FMEK, or FGAM in the FORTRAN subroutines, and functions FOMD1, FOMEK, and FOGAM are defined as

$$FO \cdots [\omega(s)] = F \cdots [\omega(s)] - s \quad .$$

Thus, in order to determine $\omega(s) > 0$ such that $\Phi[\omega(s)] = s$, we search for the value of ω such that FO ··· equals zero. Subroutine ZERO performs the search, computing the relative accuracy (change in ω from the last iteration). When the absolute value of the relative accuracy is less than a specified tolerance, the search stops, and the current value of ω is used to compute the Laplace transform using equation (2.2.4).

The choice of tolerance value (FORTRAN variable TOLRNC) affects the accuracy of ω determined from the search, and the accuracy of ω affects $p(s)$ and the estimate of $E[W(t)]$. Figure 3.5 shows both the approximations of $E[W(t)]$ based upon the exact quadratic solution of the functional equation (2.2.4) and the approximations obtained using the Newton-Raphson search with various specified tolerances. We observe that the two results agree for nine or ten signficiant digits in all cases and that the number of search iterations does not increase greatly as the tolerance is decreased. For each of the ten epochs, $\omega(s)$ is evaluated thirty-four times ($N = 34$); thus, the total number of iterations shown in Figure 3.5 applies to 340 evaluations of $\omega(s)$. Since the average number of iterations in each evaluation only increases from 4.2 to 5.3 as the tolerance decreases from 10^{-10} to 10^{-24}, we have set the search tolerance at 10^{-20}.

In Chapter 2 we discussed the stochastic process $W(t)$ in an M/G/1 queue and its expected value function. Equation (2.1.3) is restated here in a slightly different form along with the Laplace transform of $E_z[W(t)]$, equation (2.2.4):

Thus, in order to determine $\omega(s) > 0$ such that $\Phi[\omega(s)] = s$, we search for the value of ω such that FO ··· equals zero. Subroutine ZERO performs the search, computing the relative accuracy (change in ω from the last iteration). When the absolute value of the relative accuracy is less than a specified tolerance, the search stops, and the current value of ω is used to compute the Laplace transform using equation (2.2.4).

The choice of tolerance value (FORTRAN variable TOLRNC) affects the accuracy of ω determined from the search, and the accuracy of ω affects $p(s)$ and the estimate of $E[W(t)]$. Figure 3.5 shows both the approximations of $E[W(t)]$ based upon the exact quadratic solution of the functional equation (2.2.4) and the approximations obtained using the Newton-Raphson search with various specified tolerances. We observe that the two results agree for nine or ten signficiant digits in all cases and that the number of search iterations does not increase greatly as the tolerance is decreased. For each of the ten epochs, $\omega(s)$ is evaluated thirty-four times ($N = 34$); thus, the total number of iterations shown in Figure 3.5 applies to 340 evaluations of $\omega(s)$. Since the average number of iterations in each evaluation only increases from 4.2 to 5.3 as the tolerance decreases from 10^{-10} to 10^{-24}, we have set the search tolerance at 10^{-20}.

In Chapter 2 we discussed the stochastic process $W(t)$ in an M/G/1 queue and its expected value function. Equation (2.1.3) is restated here in a slightly different form along with the Laplace transform of $E_z[W(t)]$, equation (2.2.4):

FIGURE 3.5 EFFECT OF NEWTON-RAPHSON SEARCH TOLERANCE

THIS TABLE COMPARES CCLEMAN'S BESSEL FUNCTION RESULTS AT FIVE EPOCHS WITH LAPLACE INVERSION RESULTS FOR BOTH THE EXACT QUADRATIC SOLUTION OF THE FUNCTIONAL AND THE APPROXIMATE NEWTON-RAPHSON SEARCH SOLUTION OF THE FUNCTIONAL WITH VARIOUS TOLERANCES.

M/M/1 MEAN SERVER LOAC. LAMEDA = 1.0. E(S) = 0.95. Z = 0.

INVERSION OF LAPLACE TRANSFORM. STEHFEST'S METHOD. N=34

EPOCH T	SUMS OF BESSEL FUNCTIONS (COLEMAN)	QUADRATIC SOLUTION OF FUNCTIONAL	TOLERANCE FOR NEWTON-RAPHSON SEARCH SOLUTION OF FUNCTIONAL: 10**-10	10**-12	10**-14	10**-16
20.	3.9181	3.91805123011	3.91805123011	3.91805123011	3.91805123011	3.91805123011
40.	5.4651	5.46511190320	5.46511190320	5.46511190320	5.46511190320	5.46511190320
60.	6.5582	6.55818100638	6.55818100638	6.55818100638	6.55818100638	6.55818100638
80.	7.4191	7.41908998154	7.41908998154	7.41908998154	7.41908998154	7.41908998154
100.	8.1335	8.13354320450	8.13354320450	8.13354320450	8.13354320450	8.13354320450
200.		10.56878873791	10.56878873791	10.56878873791	10.56878873791	10.56878873791
300.		12.08196260537	12.08196260536	12.08196260537	12.08196260537	12.08196260537
400.		13.15027687636	13.15027687634	13.15027687639	13.15027687634	13.15027687639
800.		15.49230830916	15.49230830918	15.49230830918	15.49230830917	15.49230830920
1200.		16.56195996067	16.56195996093	16.56195996058	16.56195996092	16.56195996061
TOTAL NUMBER OF NEWTON-RAPHSON ITERATIONS TO EVALUATE 10 EPOCHS:			1423	1460	1518	1602

INVERSION OF LAPLACE TRANSFORM. STEHFEST'S METHOD. N=34

EPOCH T	SUMS OF BESSEL FUNCTIONS (COLEMAN)	QUADRATIC SOLUTION OF FUNCTIONAL	TOLERANCE FOR NEWTON-RAPHSON SEARCH SOLUTION OF FUNCTIONAL: 10**-18	10**-20	10**-22	10**-24
20.	3.9181	3.91805123011	3.91805123011	3.91805123011	3.91805123011	3.91805123011
40.	5.4651	5.46511190320	5.46511190320	5.46511190320	5.46511190320	5.46511190320
60.	6.5582	6.55818100638	6.55818100638	6.55818100638	6.55818100638	6.55818100638
80.	7.4191	7.41908998154	7.41908998154	7.41908998154	7.41908998154	7.41908998154
100.	8.1335	8.13354320450	8.13354320450	8.13354320450	8.13354320450	8.13354320450
200.		10.56878873791	10.56878873791	10.56878873791	10.56878873791	10.56878873791
300.		12.08196260537	12.08196260537	12.08196260537	12.08196260537	12.08196260537
400.		13.15027687636	13.15027687638	13.15027687638	13.15027687638	13.15027687638
800.		15.49230830916	15.49230830918	15.49230830918	15.49230830918	15.49230830920
1200.		16.56195996067	16.56195996062	16.56195996061	16.56195996064	16.56195996068
TOTAL NUMBER OF NEWTON-RAPHSON ITERATIONS TO EVALUATE 10 EPOCHS:			1716	1756	1773	1794

$$E_z[W(t)] = z - (1-p)t + E_z[I(t)] \quad ,$$

$$P_W(z,s) = \frac{z}{s} - \frac{\mu}{s^2} + \frac{e^{-\omega(s)z}}{s\ (s)} \quad ,$$

The first two terms of $P_W(z,s)$ can be inverted "by inspection," i.e., by referring to any table of function-transform pairs. Only the last term requires numerical inversion using LINV. Stehfest [1970, p. 48] cautioned the prospective user of LINV as follows: "One ought to be sure a priori that the unknown function $F(t)$ has not any discontinuities, salient points, sharp points, sharp peaks, or rapid oscillations." Recall that our unknown function, $E_z[I(t)]$ in M/G/1 queues, is a non-decreasing function and that it must be zero between $t = 0$ and $t = z$. Although it is not a discontinuous function, its slope does change abruptly at $t = z$, and we might expect some inaccuracies near that point. The top section of Figure 3.6 shows scaled approximations of $E_z[I(t)]$ evaluated at $t = z$ for M/M/1 queues with traffic intensities between .5 and 2 and scaled initial server loads between .2 and 4. If the inversion technique is accurate, then all entries in the table should be zero. We observe that there are some inaccuracies, particularly for queues with low traffic intensities. For example, the worst case is $\rho = .5$ with $z' = 1.0$, where scaled $E_z[I(t)]$ is .007 instead of zero.

Since the inaccuracies may be related to the discontinuity of the derivative of $E_z[I(t)]$ at $t = z$, our approach does not invert the third term of $P_W(z,s)$ directly. Instead, we construct a new function,

FIGURE 3.6 CORRECTIONS FOR DISCONTINUOUS FIRST DERIVATIVE

APPROXIMATIONS USING STEHFEST'S METHOD, N=34

M/M/1 SCALED MEAN CUMULATIVE IDLENESS AT T'=ABS(MU)*Z' (I.E., AT T=Z)

WITHOUT CORRECTION TERM

SCALED INITIAL SERVER LOAD

RHO	0.2	0.4	0.6	0.8	1.0	2.0	3.0	4.0
0.5	0.003122	0.005112	0.006274	0.006845	0.007000	0.005125	0.002718	0.001107
0.6	0.002661	0.003710	0.003879	0.003605	0.003140	0.000965	0.000101	-.000093
0.7	0.001984	0.002061	0.001606	0.001108	0.000707	-.000004	-.000021	0.000001
0.8	0.001059	0.000587	0.000235	0.000072	0.000010	-.000001	0.000000	-.000000
0.9	0.000148	0.000006	-.000002	-.000000	0.000000	-.000000	-.000000	-.000000
1.1	0.000029	-.000001	-.000000	0.000000	0.000000	0.000000	-.000000	0.000000
1.2	0.000214	0.000022	-.000000	-.000001	-.000000	-.000000	0.000000	0.000000
1.3	0.000401	0.000083	0.000011	0.000000	-.000001	0.000000	-.000000	0.000000
1.4	0.000538	0.000151	0.000030	0.000004	-.000000	0.000000	0.000000	-.000000
1.5	0.000631	0.000209	0.000051	0.000009	0.000001	-.000000	0.000000	-.000000
2.0	0.000771	0.000313	0.000095	0.000025	0.000006	-.000000	-.000000	0.000000

M/M/1 SCALED MEAN CUMULATIVE IDLENESS AT T'=ABS(MU)*Z' (I.E., AT T=Z)

WITH CORRECTION TERM

SCALED INITIAL SERVER LOAD

RHO	0.2	0.4	0.6	0.8	1.0	2.0	3.0	4.0
0.5	0.000002	0.000002	-.000001	-.000006	-.000010	-.000033	-.000129	-.000289
0.6	0.000002	-.000000	-.000004	-.000007	-.000010	-.000076	-.000157	-.000150
0.7	0.000001	-.000002	-.000004	-.000009	-.000020	-.000060	-.000024	0.000001
0.8	-.000001	-.000003	-.000011	-.000019	-.000021	-.000001	0.000000	-.000000
0.9	-.000001	-.000006	-.000003	-.000000	0.000000	-.000000	-.000000	-.000000
1.1	-.000002	-.000001	-.000000	0.000000	0.000000	0.000000	-.000000	0.000000
1.2	0.000000	-.000002	-.000002	-.000001	-.000000	-.000000	0.000000	0.000000
1.3	0.000000	-.000001	-.000003	-.000002	-.000001	0.000000	-.000000	0.000000
1.4	0.000001	0.000000	-.000002	-.000002	-.000001	0.000000	-.000000	-.000000
1.5	0.000001	0.000001	-.000001	-.000002	-.000001	-.000000	0.000000	-.000000
2.0	0.000002	0.000002	0.000001	-.000000	-.000001	-.000000	-.000000	0.000000

$$H(t) = E_z[I(t)] + g(t) \ ,$$

where $g(t)$ and its Laplace transform are known, and where $g(t)$ is chosen so that both $H(t)$ and its derivative are continuous. Then LINV is used to invert the Laplace transform of $H(t)$, and approximate $E_z[I(t)]$ is obtained by subtracting $g(t)$ from approximate $H(t)$.

It can be shown that the slope of $E_z[I(t)]$ in M/G/1 queues is equal to the probability that the server load is zero, denoted $P\{W(t) = 0\}$, an any epoch t. Therefore, the slope of $E_z[I(t)]$ at $t = z$ is $P\{W(z) = 0\}$, which is the probability that no arrivals will occur between $t = 0$ and $t = z$, or $e^{-\lambda z}$. For $H(t)$ to have a continuous first derivative at $t = z$, that derivative must be zero. Since the slope of $E_z[I(t)]$ is $e^{-\lambda z}$ at that point, then the slope of $g(t)$ must be $-e^{-\lambda z}$. Therefore, we define $g(t)$ as follows:

$$g(t) = \begin{cases} 0 & \text{for } 0 \leq t \leq z \\ -e^{-\lambda z}(t-z) \ , & \text{for } t > z \end{cases}$$

Referring to any table of transform-function pairs, the Laplace transform of $g(t)$ is $-\exp[-z(\lambda+s)]/s^2$. Thus the Laplace transform of $H(t)$, i.e., the transform of $E_z[I(t)] + g(t)$, is

$$\text{(3.3.1)} \qquad P_H(z,s) = \int_0^\infty e^{-st} H(t)\, dt = \frac{e^{-\omega(s)z}}{s\omega(s)} - \frac{e^{-z(\lambda+s)}}{s^2} \quad .$$

Applying LINV to $P_H(z,s)$ we obtain an approximation of $H(t)$. Using overscore to denote approximations, we compute the estimate of $E_z[I(t)]$ as follows:

$$\bar{E}_z[I(t)] = \bar{H}(t) - g(t) = \bar{H}(t) + \begin{cases} 0 & \text{for } 0 \leq t \leq z \\ e^{-\lambda z}(t-z) & \text{for } t > z \end{cases}$$

The bottom section of Figure 3.6 shows scaled $\bar{E}_z[I(t)]$ when the correction term, $g(t)$, is included in the analysis. The results are considerably improved, and since the main tables will express $E_z[W(t)]$ in hundredths or thousandths, the slight inaccuracies which remain will not affect our tabulated results.

The correction term, $g(t)$, is employed in our computations for both M/D/1 and $M/E_k/1$ queues. Equation (3.3.1) is incorporated into subprogram functions PMD1 and PMEK of the main programs. The gamma input process does not require the correction term because both $E_z[I(t)]$ and its derivative are continuous, i.e., the slope of $E_z[I(t)]$ is zero at $t = z$. The method is not applied to the Wiener process because process $W(t)$ cannot be separated into four components, $z + S(t) - t + I(t)$; thus, the third term of $P_W(z,s)$ cannot be interpreted as the Laplace transform of expected cumulative idleness in this case.

So far, our discussions in this section have covered some of the important details of the two main computer programs which produce the tables of scaled $E_z[W(t)]$ is Appendix C. Listings of these two main

programs, one for $\rho = 1$ and one for $\rho \neq 1$, are included in Appendix B. We now discuss the algorithm for $M/E_k/1$ queues with $\rho \neq 1$, and we briefly mention the differences in the algorithms for the other three processes and for $\rho = 1$. Figure 3.7 describes both the steps of the algorithm and the names of the subprograms which perform those steps.

The first program specifies the values of the parameters for subroutines LINV and ZERO. Values of ρ, scaled initial server load (z^*), and sixteen scaled epochs (t^*) are read from punched cards. The program performs computations for three pages of tables on each run. The three tables on a page have the same value of ρ and epochs t^*, but each table has a different value of z^*. For each table we compute $\bar{E}_z[W(t)]$ for the Wiener process at all sixteen epochs, then for the M/D/1 queue at the same sixteen epochs, followed by each of the seven $M/E_k/1$ queues and the gamma input process.

For each of the two processes the first step is conversion of z^* and t^* to unscaled values. As discussed in Section 2.3, the time scale conversion factor is $(1-\rho)^2/\sigma^2$; this factor is denoted ALFSQR in the subprograms RZWNR, RZMD1, RZMEK, and RZGAM. The value of σ^2 for the unscaled process is selected so that the formulas and numerical computations are subsequently simplified. The conversion factor for server load or wait is $|\mu|/\sigma^2$, denoted WTFCTR in the sub-programs.

The initial trial value of $\omega(s)$ in the Newton-Raphson search for the solution of $\Phi[\omega(s)] = s$ is denoted OSTART. For a specific z^* and $\rho \leq 1$, OSTART is arbitrarily set equal to 1 for the first

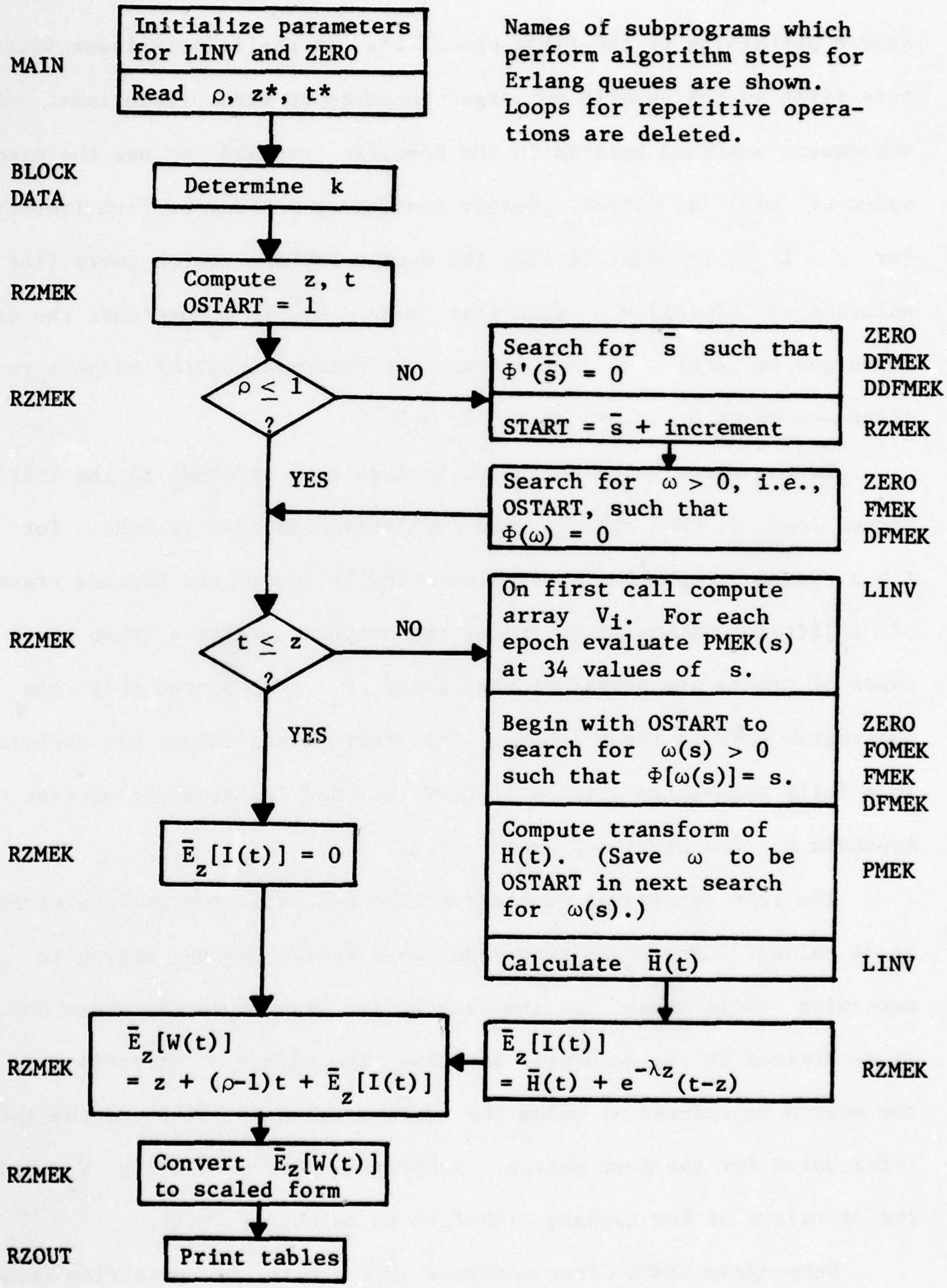


FIGURE 3.7. Flowchart of Algorithm for Tables

search pertaining to the first epoch t^*. In preliminary investigations this first search usually converged in seven or eight iterations. All subsequent searches related to the specific z^* and ρ use the previous value of $\omega(s)$ as OSTART, usually converging in four or five iterations. For $\rho > 1$ it is possible that the Newton-Raphson search could find a solution of $\Phi[\omega(s)] = s$ such that $\omega(s) < 0$. To ensure that the search converges to $\omega(s) > 0$ in this case, we determine OSTART using a two-stage search as described in Figure 3.7.

If the epoch to be evaluated is less than or equal to the initial server load z, then the expected cumulative idleness is zero. For $t > z$ we use subprogram LINV to numerically invert the Laplace transform of $E_z[I(t)]$. On any given run of the computer programs (when three pages of tables are prepared), the array V_i is computed only when subprogram LINV is first called. This feature and others are explained in a fully documented version of LINV included in the first section of Appendix B, "Computer Program for Figure 3.2."

The last section of subprogram LINV evaluates the Laplace transform at 34 values. Each evaluation requires a Newton-Raphson search to determine $\omega(s)$, except for the case of the Wiener process where OMEGA is determined by the quadratic formula. The number of iterations in the search is reduced by using the current value of $\omega(s)$ as the initial trial value for the next search. Subprogram LINV uses array V_i and the 34 values of the Laplace transform to calculate $\bar{H}(t)$.

Subprogram RZMEK first combines $\bar{H}(t)$ with the correction term to obtain $\bar{E}_z[I(t)]$, and then it computes $\bar{E}_z[W(t)]$ for the unscaled

process. In the final step $\bar{E}_z[W(t)]$ is converted to scaled form and stored in the three-dimensional array SCLWT. When the computations for three tables (each with ten processes evaluated at sixteen epochs) are completed, subprogram RZOUT prints five copies of the tables with varying margins. Then the main program reads punched cards specifying ρ, z^*, and t^* for the next page of tables.

All computer programs listed in Appendix B were coded in IBM FORTRAN IV [IBM, 1971] and processed on an IBM 370/168 with the high-speed-multiply feature. The average computer time needed to evaluate scaled $E_z[W(t)]$ for one process at one epoch, i.e., a single value in the tables of Appendix C, was approximately 0.17 second.

CHAPTER 4

NUMERICAL RESULTS FOR EXPECTED SERVER LOAD

4.1. Using the Tables

In this section we explain how the tables in Appendix C can be used to obtain time-dependent mean server load in $M/E_k/1$ queues. We will repeat only the notation and formulas from the previous chapters that are required for using the tables; the details of the underlying theory and numerical method will not be discussed here.

The system being studied is the standard $M/E_k/1$ queue. We assume that the analyst has specified the value of the mean arrival rate, denoted λ, and has determined that the service times (random variable S) can be described by a gamma or Erlang distribution. If an actual operating system is being studied and data are available, the analyst can use the methods described by Hora [1978] or Reinmuth [1971] to verify that the input process is Poisson. Similarly, a chi-square goodness-of-fit test can be used to compare the actual distribution of service times with tabulated values of the incomplete gamma function [Pearson, 1922]. In cases where the service times of an actual system do not exactly fit the gamma distribution, the analyst still may wish to use this theoretical distribution as an approximation.

We define the probability density function for this two-parameter gamma distribution using the mean, E(S), and the Erlang shape parameter, k (not restricted to integer values), as follows:

$$f(t) = \frac{[k/E(S)]^k}{(k-1)!} t^{k-1} e^{-kt/E(S)} , \qquad t \geq 0,\ k > 0 .$$

The standard deviation of this service time distribution is $E(S)/\sqrt{k}$, so the appropriate value of parameter k can be determined either from the mean and second moment, $E(S^2)$, or from the mean and variance, $Var(S)$, as follows:

$$k = \frac{[E(S)]^2}{Var(S)} = \frac{[E(S)]^2}{E(S^2) - [E(S)]^2} . \tag{4.1.1}$$

Figure 4.1 shows two groups of gamma service time distributions, where the special case of the exponential distribution $(k = 1)$ is included in both groups. The mean, $E(S)$, of each distribution in the figure is equal to unity, so the standard deviation is $1/\sqrt{k}$ in each case. The top chart includes density functions with $k = 2$ and $k = 4$, i.e., with less variance than the exponential distribution. The limiting case as k becomes infinitely large is the degenerate distribution with constant service times, corresponding to the M/D/1 queue. The bottom chart of Figure 4.1 includes density functions with $k = .5$ and $k = .2$; these two distributions can be used to approximate systems where the service times have more variability than the exponential case.

The stochastic process being studied is server load, denoted $W(t)$. This process is also called unfinished work, system backlog, or virtual waiting time, the latter because $W(t)$ represents the time that an arrival at epoch t would wait before beginning service. The system may begin operation at epoch $t = 0$ with some initial backlog, $W(0) = z$.

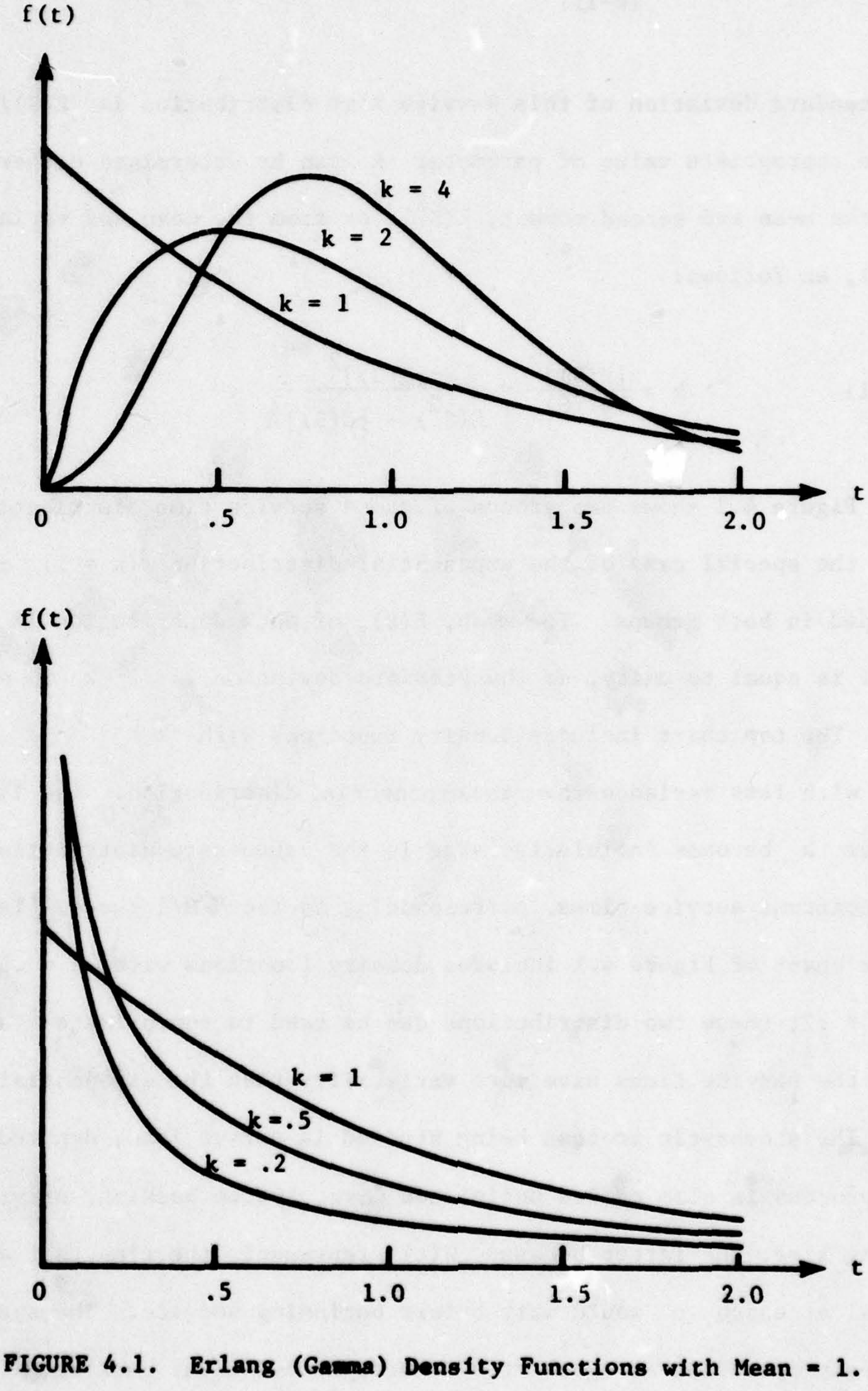


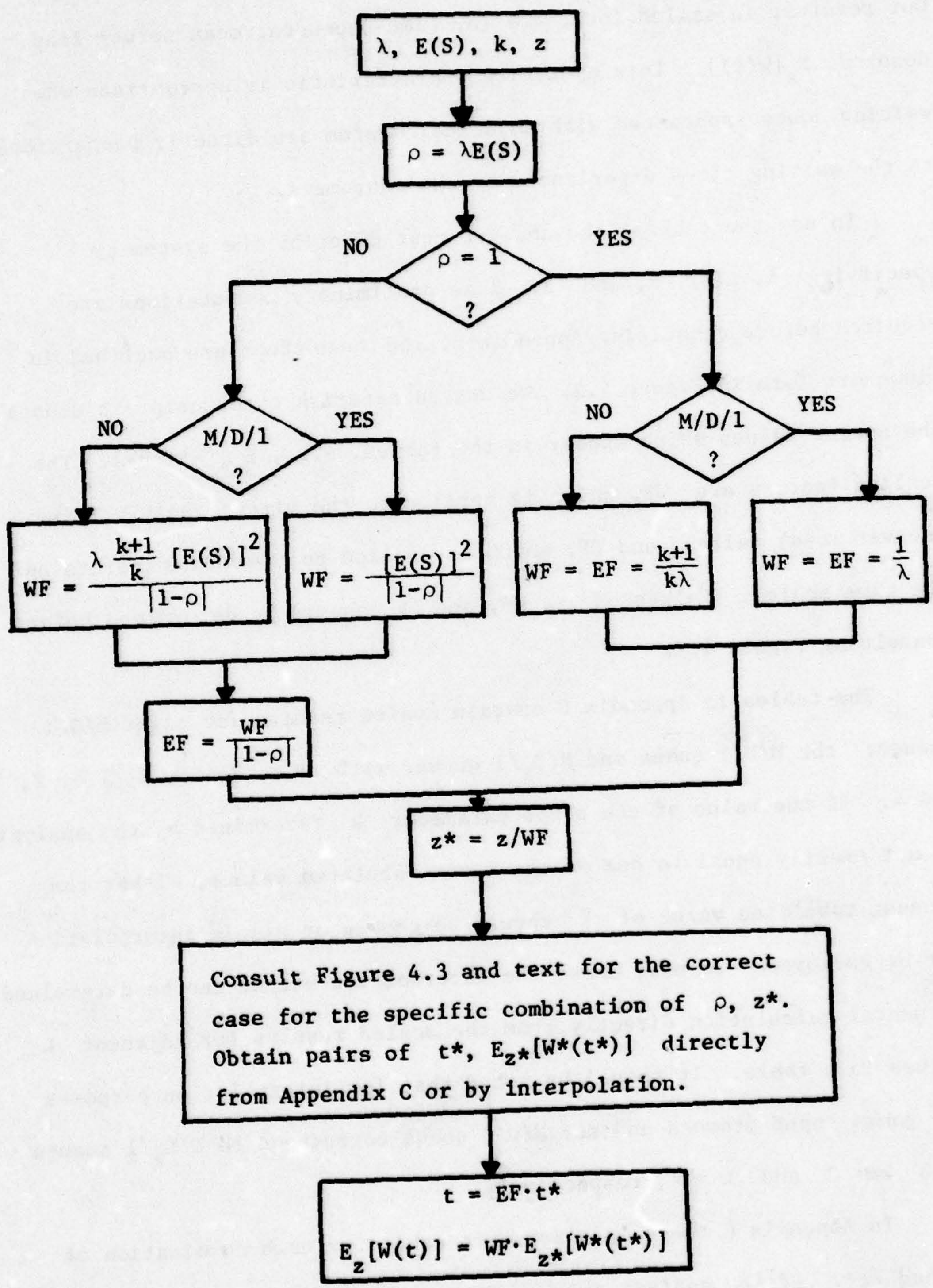
FIGURE 4.1. Erlang (Gamma) Density Functions with Mean = 1.

Our results, in scaled form, are for time-dependent mean server load, denoted $E_z[W(t)]$. This operating characteristic is appropriate when waiting costs associated with an actual system are directly proportional to the waiting times experienced by the customers.

To use the tables, the analyst must describe the system by specifying λ, $E(S)$, k, and z. Some preliminary computations are required before consulting Appendix C, and these steps are outlined in flowchart form in Figure 4.2. We use an asterisk superscript to denote the scaled values which appear in the tables, e.g., $E_{z*}[W*(t*)]$. The scaling factors are WF, which is applied to the virtual waiting time (server load) values, and EF, which is applied to the epochs (points on the time scale). Values of ρ, $z*$, and k should be determined before consulting Figure 4.3.

The tables in Appendix C contain scaled results for eight M/G/1 queues: the M/D/1 queue and $M/E_k/1$ queues with $k = .2, .5, 1, 1.5, 2, 3,$ and 4. If the value of the shape parameter k determined by the analyst is not exactly equal to one of the seven tabulated values, either the closest tabulated value of k should be chosen or simple interpolation can be employed. In most cases the interpolated values can be determined by mental calculation directly from the scaled results for adjacent k values in a table. It should be noted that for interpolation purposes the gamma input process and the M/D/1 queue correspond to $M/E_k/1$ queues with $k = 0$ and $k = \infty$, respectively.

In Appendix C there is a separate table for each combination of ρ and $z*$. If the analyst chooses not to use the closest $(\rho, z*)$



Note: EF = Epoch Factor for scaling

WF = Wait Factor for scaling

FIGURE 4.2. Flowchart Instructions for Using the Tables

	$z^* = 0, .2, .4, .6, .8, 1.0$	$0 < z^* < 1$; $z^* \neq .2, .4, .6, .8$	$z^* = 2., 3., 4.$	$1 < z^* < 4$; $z^* \neq 2., 3.$	$z^* > 4.$
$\rho < .5$	I	I	I	I	I
$\rho = .5, .6, .7, .8, .9$	A	C	A	C	H
$.5 < \rho < .9$; $\rho \neq .6, .7, .8$	B	D	B	D	H
$\rho = 1.0$	A	C	A	C	G
$\rho = 1.1, 1.2, 1.3, 1.4, 1.5, 2.$	A	C	F	F	F
$1.1 < \rho < 2.$; $\rho \neq 1.2, 1.3, 1.4, 1.5$	B	D	F	F	F
$\rho > 2.0$	E	E	F	F	F

FIGURE 4.3. Various Cases for (ρ, z^*) Combinations

combination which is tabulated, then interpolation between two or more tables will be required. Figure 4.3 illustrates the cases which might be encountered, and the appropriate methods for these cases are discussed below. For any specific (ρ, z^*) combination being studied, the objective is to determine pairs of t^* and $E_{z^*}[W^*(t^*)]$ to which the scaling factors, EF and WF, will be applied.

Case A. No interpolation is required. There is a table in Appendix C for this specific (ρ, z^*) combination, and the values for scaled epochs, t^*, and scaled mean server load, $E_{z^*}[W^*(t^*)]$, can be read directly from the table.

Case B. The exact value of z^* is tabulated, but the exact value of ρ falls between two tabulated values. First find the two tables that have the exact z^* with the two closest values of ρ. Then list the scaled epochs (t^*) that appear on both tables, and also list both values of $E_{z^*}[W^*(t^*)]$ for each of the common epochs. For each epoch use simple interpolation to determine $E_{z^*}[W^*(t^*)]$ for the exact value of ρ.

Case C. The exact value of ρ is tabulated, but the exact value of z^* falls between two tabulated values. The method is similar to Case B. Find the two tables with the exact ρ and the two closest values of z^*; list the common values of t^* with the two values of $E_{z^*}[W^*(t^*)]$; and use simple interpolation. The initial behavior of

the system can also be determined as follows:

$$E_{z^*}[W^*(t^*)] = \begin{cases} z^* - t^* & \text{for } \rho < 1,\ 0 \le t^* \le (1-\rho)z^* \\ z^* & \text{for } \rho = 1,\ 0 \le t^* \le z^* \\ z^* + t^* & \text{for } \rho > 1,\ 0 \le t^* \le (\rho-1)z^* \end{cases}$$

Case D. The exact values of neither ρ nor z^* are tabulated, but the exact value of each does fall between two tabulated values. For one of the closest z^* values which are tabulated, perform the interpolation between the two closest values of ρ using the procedure of Case B. Use the same procedure for the other closest value of z^*. Finally, use the procedure of Case C to interpolate between those two sets of data. For example, if we are interested in results for ($\rho = .67$, $z^* = .44$), we interpolate between ($\rho = .6$, $z^* = .4$) and ($\rho = .7$, $z^* = .4$) to obtain ($\rho = .67$, $z^* = .4$). We also interpolate between ($\rho = .6$, $z^* = .6$) and ($\rho = .7$, $z^* = .6$) to obtain ($\rho = .67$, $z^* = .6$). Finally, we interpolate between ($\rho = .67$, $z^* = .4$) and ($\rho = .67$, $z^* = .6$) to obtain results for ($\rho = .67$, $z^* = .44$).

Case E. For situations where $\rho > 2$ and $0 < z^* < 1$, the Wiener process provides an upper bound for $E_{z^*}[W^*(t^*)]$ in any queueing system. Because of the scaling used in the tables, the tabulated values of $E_{z^*}[W^*(t^*)]$ for the Wiener process depend only on z^*, not on ρ. Therefore, any table with $\rho \geq 1.1$ and the specified value of z^* can be used to determine the upper bound. Interpolation can be used if the

exact value of z^* falls between the tabulated values. In these same situations a lower bound for $E_{z^*}[W^*(t^*)]$ is $z^* + t^*$.

<u>Case F</u>. For situations with both $\rho > 1$ and $z^* > 1$, a very accurate approximation of $E_{z^*}[W^*(t^*)]$ is $z^* + t^*$. The accuracy is best when both ρ and z^* are large, but even for $\rho = 1.1$ and $z^* = 1$ the maximum error (M/D/1 queue, $t^* = 1.5$) is only 1.4 percent.

<u>Case G</u>. For $\rho = 1$ and $z^* > 4$, the initial behavior is $E_{z^*}[W^*(t^*)] = z^*$ for $0 \leq t^* \leq z^*$. For $t^* > z^*$, the results for $z^* = 4$ provide a loose lower bound.

<u>Case H</u>. For $.5 \leq \rho \leq .9$ and $z^* > 4$, the initial behavior is $E_{z^*}[W^*(t^*)] = z^* - t^*$, for $0 \leq t^* \leq (1-\rho)z^*$. For $t^* > (1-\rho)z^*$, the results for $z^* = 4$ provide a lower bound.

<u>Case I</u>. For systems with $\rho < .5$ the approach to equilibrium is very slow. If the initial server load $z^* > 0$, the initial behavior is the same as Case H. For $z^* \leq 4$ the results for the same queue (i.e., the same value of k) with $\rho = .5$ provide an upper bound.

<u>Other Cases</u>. For situations with $.9 < \rho < 1.0$ and $0 \leq z^* \leq 4$, the tabulated results for the Wiener process with any $\rho < 1$ and the appropriate z^* provide a tight upper bound for any queueing system. Likewise, for situations with $1.0 < \rho < 1.1$ and $0 \leq z^* \leq 4$, the

tabulated results for the Wiener process with any $\rho > 1$ and the appropriate z^* provide a tight upper bound. In both cases the Wiener approximation is most accurate for values of ρ closest to unity.

The tables in Appendix C can be used along with Figures 4.2 and 4.3 to obtain explicit results for the transient behavior of mean server load in a wide range of M/G/1 queues. In many applications the analyst will perform some sensitivity analysis by using various estimates of λ, E(S), k, and z and comparing the results. In cases where $\rho < 1$ the analyst may be interested only in the speed with which the system approaches steady-state. Expressing the Pollaczek-Khintchine formula with our notation, the mean of the steady-state distribution for server load (often called expected waiting time in the queue, excluding service) is

$$E_z[W(\infty)] = WF/2 , \qquad z \geq 0,\ \rho < 1 .$$

4.2. Several Sample Problems

In this section we illustrate the methods described in Section 4.1 by working several sample problems. The first three problems show how changes in the variance of service times affect the transient behavior of mean server load and the steady-state value, assuming the other parameters of the system are held constant. The fourth and fifth problems illustrate using interpolation to determine results for queues when the exact value of ρ or z^* is not tabulated. The determination of waiting

costs using time-dependent mean server load is discussed and applied to one of the sample problems.

The following problems are adapted from an example described by Hillier and Lieberman [1974, p. 436]. A university plans to lease a small batch-processing computer for its students to use. It is expected that the students will submit programs to be run every 3 minutes on the average and that the times between submission of programs have an exponential distribution. The computer under consideration could process an average of 25 typical student programs per hour if it were run continuously. We will examine the transient behavior of this M/G/1 system for an eight-hour period, assuming that the system begins operation with no backlog. Thus far we have specified λ, E(S), and z. For Problem A we assume that the processing times for the students' programs are exponentially distributed, i.e., $k = 1$. Referring to Figure 4.2, we perform the initial computations, using hours as the time unit.

<u>Problem A.</u>

$$\lambda = 20,\ E(S) = 1/25,\ k = 1,\ z = 0$$

$$\rho = 20\cdot(1/25) = .8$$

$$WF = \frac{20\cdot 2\cdot (1/25)^2}{1-.8} = .32$$

$$EF = .32/.2 = 1.6$$

$$z^* = 0/.32 = 0$$

Referring to Figure 4.3 with $\rho = .8$ and $z^* = 0$, we see that Case A is appropriate, i.e., the exact values of ρ and z^* appear in a table in Appendix C. Referring to the appropriate table, we note that the sixteen scaled epochs t^* are between .02 and 4. Eight points should be sufficient to describe the queue's behavior; the selected values of t^* and $E_{z^*}[W^*(t^*)]$ from Appendix C are shown below in the first two columns. Referring to Figure 4.2, we compute t and $E_z[W(t)]$ by multiplying the scaled values times the scaling factors. The desired results are shown below in the last two columns.

Problem A.

t^*	$E_{z^*}[W^*(t^*)]$	t	$E_z[W(t)]$
.1	.164	.16	.052
.2	.230	.32	.074
.4	.306	.64	.098
.6	.351	.96	.112
1.	.405	1.6	.130
2.	.461	3.2	.148
3.	.482	4.8	.154
4.	.491	6.4	.157

The steady-state mean server load in Problem A is $WF/2 = .16$ hour (9.6 minutes), or four average service times. After approximately one hour of operations ($t = .96$), mean server load (.112 hour) in this M/M/1 queue is about 70% of steady-state; after three hours it achieves

90% of the steady-state value. For queues with $\rho < 1$, $E_{z^*}[W^*(t^*)]$ approaches a steady-state value of .5; therefore, the percentage of steady-state achieved at any epoch can be determined simply by doubling the $E_{z^*}[W^*(t^*)]$ value.

For Problem B we consider a system with the same arrival rate and the same mean service time, but we assume that the service times are constant (deterministic). In the context of our original computer-leasing problem, it might be that the student programs are actually uniform or that this specific computer has very little variation in processing times for the programs being run. Referring to Figure 4.2, we perform the computations for this M/D/1 queue.

Problem B.

$$\lambda = 20,\ E(S) = 1/25,\ k = \infty,\ z = 0$$

$$\rho = 20\cdot(1/25) = .8$$

$$WF = \frac{20\cdot(1/25)^2}{1-.8} = .16$$

$$EF = .16/.2 = .8$$

$$z^* = 0/.16 = 0$$

Referring to Figure 4.3, we see that the exact values of ρ and z^* are tabulated, and we select eight points from the appropriate table in Appendix C as shown below. The tabulated values are mutiplied by EF and WF to obtain t and $E_z[W(t)]$, respectively.

Problem B.

t^*	$E_{z^*}[W^*(t^*)]$	t	$E_z[W(t)]$
.2	.242	.16	.039
.4	.316	.32	.051
.6	.360	.48	.058
1.	.412	.8	.066
1.5	.446	1.2	.071
2.	.465	1.6	.074
3.	.484	2.4	.077
4.	.492	3.2	.079

The steady-state mean server load in Problem B is $WF/2 = .08$ hour (4.8 minutes), or two service times. We observe that after a half-hour of operation ($t = .48$) mean server load (.058 hour) is about 70% of steady-state; after one and a half hours ($t = 1.6$), it has reached 90% of the steady-state value (.074 relative to .08). Compared with the M/M/1 queue of Problem A, this M/D/1 queue has a lower steady-state mean server load and it approaches that equilibrium value much faster.

For Problem C we consider a system with the same prameters as Problems A and B, except that the service times have wide variability. We assume that the mean service time is 2.4 minutes (1/25 hour), as before, but that the standard deviation of the service times is 5.4 minutes. Referring to equation (4.1.1), the appropriate Erlang shape

parameter is $k = .2$. The computations from Figure 4.2 and the selected points from Appendix C are shown below.

Problem C.

$$\lambda = 20,\ E(S) = 1/25,\ k = .2,\ z = 0$$

$$\rho = 20\cdot(1/25) = .8$$

$$WF = \frac{20\cdot 6\cdot (1/25)^2}{1-.8} = .96$$

$$EF = .96/.2 = 4.8$$

$$z^* = 0/.96 = 0$$

t^*	$E_{z^*}[W^*(t^*)]$	t	$E_z[W(t)]$
.1	.156	.48	.150
.2	.223	.96	.214
.4	.299	1.92	.287
.6	.345	2.88	.331
.8	.377	3.84	.362
1.	.400	4.8	.384
1.5	.438	7.2	.420
2.	.459	9.6	.441

The steady-state mean server load in Problem C is $WF/2 = .48$ hour (28.8 minutes), or twelve average service times, which is much higher than the M/M/1 and M/D/1 cases. We observe that about 70% of steady-state is achieved after three hours and that approximately eight hours of operation are needed to reach 90% of the steady-state value.

Figure 4.4 shows mean server load curves for Problems A, B, and C, where these three queues have the same values of λ and E(S). We observe that the queues with higher variance of service times, i.e., with lower values of k, have a higher steady-state mean server load. Furthermore, the approach to equilibrium is slower for the queues with the higher steady-state value. We also note from Figure 4.4 that the ordering of the curves is the opposite of the ordering of the scaled values in the table ($\rho = .8$, $z^* \approx 0$) in Appendix C; this reordering is explained by the different scaling factors, WF and EF, for the three queues.

Returning to our computer-leasing problem, we next consider a computer that can process an average of 30 programs per hour, i.e., $E(S) = 1/30$. We assume that the processing times are exponentially distributed, i.e., $k = 1$, and that the arrival pattern is the same as Problems A, B, and C. For Problem D we will assume that students can leave programs during the night for processing when the facility opens. It is estimated that "about a dozen" programs will arrive during a typical night; thus, the initial server load (assuming 12 jobs at 1/30 hour each) is .4 hour. Referring to Figure 4.2, we perform the following calculations.

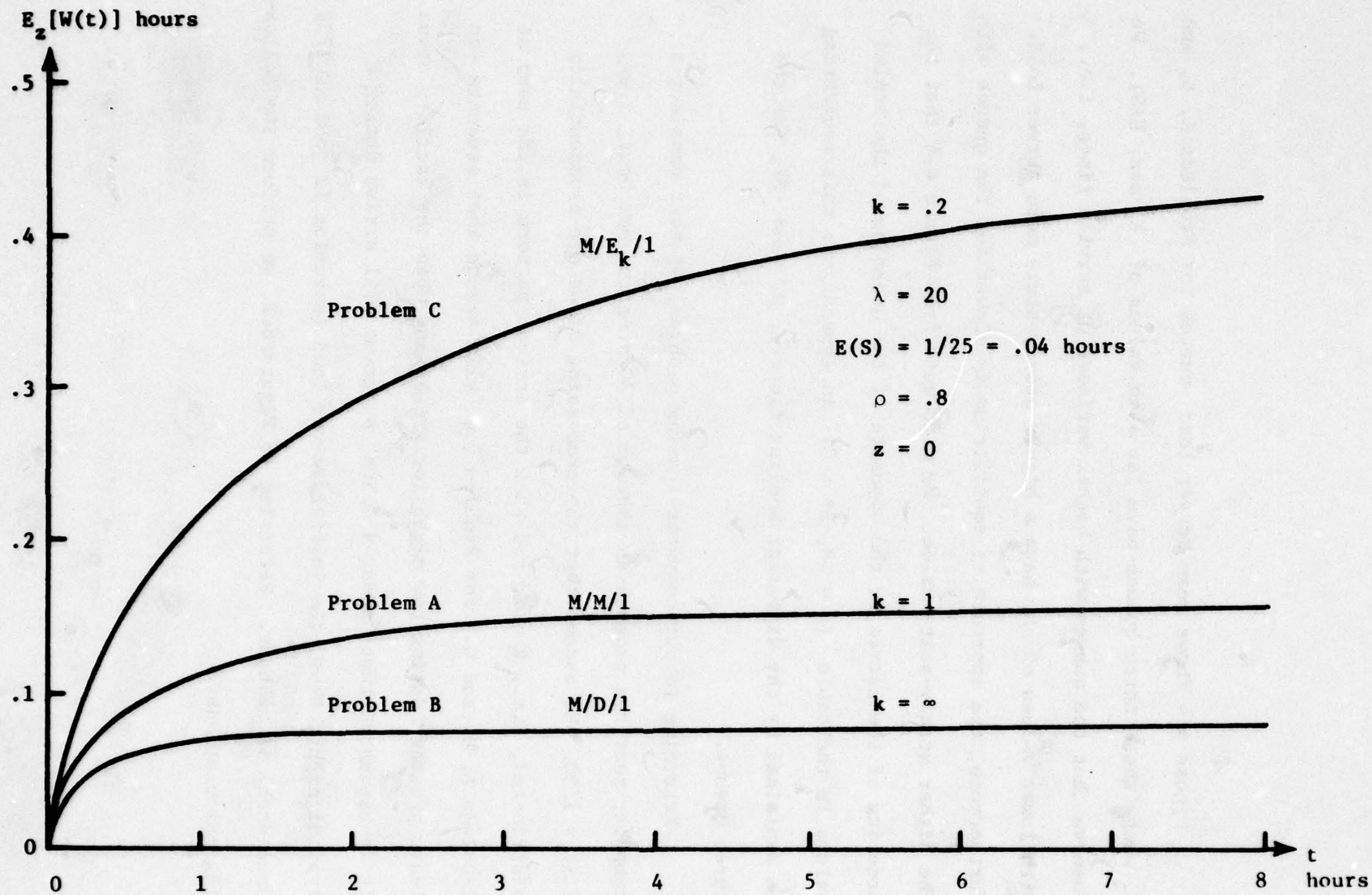


FIGURE 4.4. Graphs for Sample Problems A, B, and C.

Problem D.

Values from Appendix C

	$E_{z^*}[W^*(t^*)]$, $z^* = 3$		Interpolated $E_{z^*}[W^*(t^*)]$		$E_z[W(t)]$
t^*	$\rho = .6$	$\rho = .7$	$\rho = .67$	t	$\rho = .67$
.5	2.5	2.5	2.5	.2	.333
1.	2.0	2.0	2.0	.4	.267
1.5	1.528	1.551	1.543	.6	.206
2.	1.195	1.225	1.215	.8	.162
2.5	.978	1.004	.995	1.	.133
3.	.835	.854	.848	1.2	.113
4.	.673	.681	.678	1.6	.090
5.	.593	.596	.595	2.	.079
6.	.552	.552	.552	2.4	.074
7.	.530	.529	.529	2.8	.071
8.	.517	.517	.517	3.2	.069
10.	.506	.506	.506	4.	.067
12.	.502	.502	.502	4.8	.067

Problem D.

$$\lambda = 20,\ E(S) = 1/30,\ k = 1,\ z = .4$$

$$\rho = 20\cdot(1/30) = .67$$

$$WF = \frac{20\cdot 2\cdot (1/30)^2}{1-.67} = .133$$

$$EF = .133/.333 = .4$$

$$z^* = .4/.133 = 3$$

Referring to Figure 4.3 with $\rho = .67$ and $z^* = 3$, we see that Case B is appropriate. The two relevant tables are $(\rho = .6,\ z^* = 3)$ and $(\rho = .7,\ z^* = 3)$, and we list the values of t^* appearing on both tables. Those thirteen values and the corresponding values of $E_{z^*}[W^*(t^*)]$ from the two tables are shown in the first three columns below. For each scaled epoch we interpolate to find the approximate value that is two-thirds (or about seven-tenths) of the difference between the values for $\rho = .6$ and $\rho = .7$. The interpolated value is shown below in the fourth column. Finally, we obtain t and $E_z[W(t)]$ by multiplying t^* and the interpolated values of $E_{z^*}[W^*(t^*)]$ times EF and WF, respectively.

The steady-state mean server load in Problem D is $WF/2 = .067$ hour (4 minutes), or two mean service times. We observe that transient mean server load is within 10% of the steady-state value after approximately two and a half hours of operation. Figure 4.5 shows the curve for Problem D and the curve for the same queue with no initial load. (The calculations for the latter case are not shown here.) The system with $z = 0$ reaches 90% of steady-state after only .8 hour of operation.

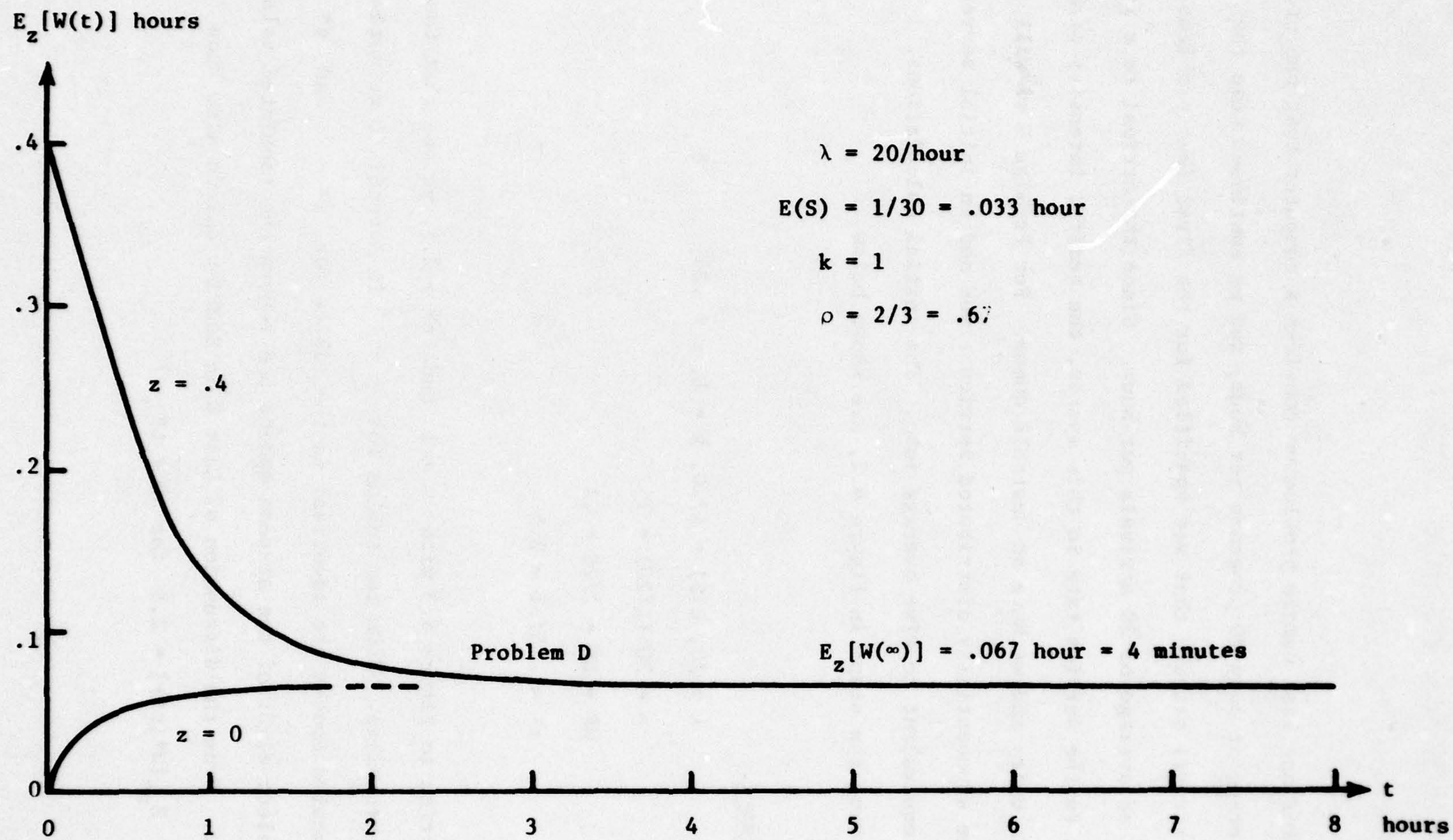


FIGURE 4.5. Graph for Sample Problem D.

For our last sample problem we consider a computer that can process an average of only 20 programs per hour, and we continue to use the same arrival pattern that was specified for the first four problems, i.e., an average of 20 arrivals per hour. Since the arrival rate is equal to the service rate in this system, the traffic intensity parameter equals unity and we have an unstable queue. For Problem E we will assume exponentially distributed service times and an initial server load equivalent to five average jobs. The initial calculations, following the steps in Figure 4.2, are shown below.

Problem E.

$$\lambda = 20,\ E(S) = 1/20,\ k = 1,\ z = .25$$

$$\rho = 20\cdot(1/20) = 1$$

$$WF = EF = 2/20 = .1$$

$$z^* = .25/.1 = 2.5$$

Referring to Figure 4.3 with $\rho = 1$ and $z^* = 2.5$, we see that Case C is appropriate. From the tables for $\rho = 1$ in Appendix C we note that the scaled epochs are identical in the tables for $z^* = 2$ and $z^* = 3$. We select eight of the sixteen epochs and record the tabulated values below. From the discussion of Case C in Section 4.1 we also know that $E_{z^*}[W^*(t^*)] = 2.5$ for $0 \leq t^* \leq 2.5$.

Problem E.

Values from Appendix C

			Interpolated		
	$E_{z*}[W*(t*)]$, $\rho = 1$		$E_{z*}[W*(t*)]$		$E_z[W(t)]$
t*	z* = 2	z* = 3	z* = 2.5	t	z = .25
0 to 2.5			2.5	0 to .25	.25
3	2.056	3.000	2.528	.3	.253
4	2.149	3.016	2.583	.4	.258
6	2.362	3.108	2.735	.6	.274
8	2.577	3.238	2.908	.8	.291
10	2.784	3.383	3.084	1.	.308
20	3.680	4.115	3.898	2.	.390
40	5.052	5.364	5.208	4.	.521
80	7.068	7.289	7.179	8.	.718

If the M/M/1 system described in Problem E continues operating indefinitely, the mean server load will increase without bound. In many actual systems the arrival rate equals or exceeds the service rate for short periods of time, but such situations may be acceptable if the cost of providing faster service exceeds the cost associated with the high waiting times.

The five sample problems show that the methods described in this chapter can be used to determine time-dependent mean virtual waiting time, an important operating characteristic of M/G/1 queues. In actual applications the analyst may wish to determine the cost

associated with $E_z[W(t)]$ when analyzing or designing a system. In their discussion of decision models for queueing systems, Hillier and Lieberman [1974, Ch. 10] discuss appropriate cost functions for various problems and apply their methods to steady-state distributions of both the number of customers in the system and the waiting times of individual customers. As previously mentioned, our results for expected server load are appropriate for evaluating costs only in systems where the waiting costs are directly proportional to waiting times. We next describe a method for evaluating the waiting cost in situations where the transient behavior of the system is important.

Let c be the cost per unit of waiting time experienced by an individual customer. The cost associated with a customer arriving at epoch t is $c \cdot W(t)$, where there may be some initial server load $W(0) = z$ for process $W(\cdot)$. For any specific epoch t, $W(t)$ is a random variable; in our situation the cost function is linear, and the expected cost for a customer arriving at epoch t is $c \cdot E_z[W(t)]$. In cases where the cost function is non-linear, we would need more information about the distribution of $W(t)$, not just its mean, in order to determine the expected cost of a customer arriving at epoch t. The probability that a customer will arrive during interval $(t, t+\Delta t)$ is λdt, and such a customer would incur expected cost $c \cdot E_z[W(t)]$. Thus, the expected waiting cost during a period from $t = 0$ to $t = T$ is

$$\int_0^T c\, E_z[W(t)]\, \lambda dt = c\lambda \int_0^T E_z[W(t)]\, dt \quad . \tag{4.2.1}$$

Note that the integral on the right-hand-side of (4.21) is simply the area under the curve for transient mean server load.

For a queueing system that approaches steady-state quite slowly, the analyst should use the integral expression, not the equilibrium mean waiting time, to evaluate expected cost. To illustrate the difference, consider again the system described in Problem C. Assume the university has determined that for this situation the appropriate cost associated with making a student wait for a program to be processed is $10 per hour. If we approximate the mean waiting time using the steady-state value of .48 hours, then the expected waiting cost for an individual student arriving with a computer program to be run is $4.80. The mean arrival rate is 20 per hour, so the expected waiting cost is $96 for each hour the system is in operation. Thus, during an eight-hour day the total expected waiting cost is $768. This total cost figure is based on the equilibrium mean waiting time, but from Figure 4.4 we can see that $E_z[W(t)]$ for Problem C is substantially less than the equilibrium value.

For a more accurate determination of total waiting cost during the eight-hour period, we can evaluate the integral on the right-hand-side of equation (4.2.1) graphically by plotting the curve from Figure 4.4 on a fine grid and counting the squares. Using this method the value of the area under the curve is approximately 2.66 hours^2. (The corresponding value using the equilibrium mean server load is $.48 \cdot 8 = 3.84$ hours^2.) Multiplying the area under the curve by $c \cdot \lambda$, we find that the correct total expected waiting cost for the eight-hour period is $\$10 \cdot 20 \cdot 2.66 = \$532$.

In this particular problem the use of the equilibrium value produces a total cost figure approximately 44% greater than the correct amount. The example illustrates the importance of using time-dependent results when the approach to equilibrium is slow. The graphical technique can also be used to determine expected waiting cost for unstable queueing systems ($\rho \geq 1$), where all behavior is transient and no equilibrium is ever achieved.

CHAPTER 5

CONCLUSIONS

The objective of this research was to provide tables for time-dependent mean server load in M/G/1 queueing systems. Chapter 2 presented the theoretical framework for the server load process, the Laplace transform of $E_z[W(t)]$, and a scaling procedure that allowed us to reduce the number of parameters required to describe a specific system from four $[\lambda, E(S), z, k]$ to three $[\rho, z^*, k]$. In Chapter 3 we considered a technique for inverting the Laplace transform, and we conducted extensive checks and comparisons to ensure the accuracy of our numerical results. The sample problems in Chapter 4 illustrated that a practitioner can easily use our tabulated results in Appendix C by following simple step-by-step procedures. We now discuss some additional results of this research, including a study of the error associated with the Brownian approximation, and we offer some suggestions for future research.

The scaling procedure employed in this research produces curves for $E_{z^*}[W^*(t^*)]$ that are monotonically ordered by each of the three parameters (ρ, z^*, and k), and these consistent orderings facilitated the interpolation procedures described in Chapter 4. Referring to any single table in Appendix C, i.e., fixing the values of ρ and z^*, we observe that the curves for $E_{z^*}[W^*(t^*)]$ increase monotonically as k, the shape parameter of the Erlang service time distribution, increases. (This ordering of the transient curves by the shape parameter has not been proven analytically and is a possible topic for future theoretical

research.) As we would expect, the upper bound (corresponding to $k = \infty$) is the curve for the M/D/1 queue. But our numerical results also indicate that a lower bound can be identified: the gamma input process (corresponding to $k = 0$). We can think of these upper and lower bounds as defining an "envelope", and we might hypothesize that this envelope contains the curves, in scaled form, for all M/G/1 queueing systems, not just for the $M/E_k/1$ queues that are studied here. Figure 5.1 shows the upper and lower bounds for the ($\rho = .5$, $z^* = 0$) case. We observe that the envelope is relatively narrow; specifically, its maximum width is 11% of the steady-state value in this case. For $z^* = 0$ and $\rho = .6, .7, .8$, and $.9$, the maximum width is 9%, 7%, 5%, and 3%, respectively. We could use these bounds to obtain approximate results for any M/G/1 queueing system for which λ, $E(S)$, $E(S^2)$, and z are known, without specifying the exact shape of the service time distribution.

Another monotonic ordering is observed by fixing the values of k and z^* and varying ρ. For example, if we examine the curves for the M/D/1 queue ($k = \infty$) with $z^* = 0$, we observe that for $\rho < 1$ the scaled curves increase monotonically as ρ increases. (In Appendix C this comparison requires examining tables that are each three pages apart.) The M/D/1 curves seem to be approaching the curve for the Wiener process as ρ increases. This observation has not been proven analytically, but the heavy traffic (or Brownian motion) approximation for the equilibrium distribution of M/G/1 server load is derived by examining, the scaled form, the limit of a sequence of queues as ρ

research.) As we would expect, the upper bound (corresponding to $k = \infty$) is the curve for the M/D/1 queue. But our numerical results also indicate that a lower bound can be identified: the gamma input process (corresponding to $k = 0$). We can think of these upper and lower bounds as defining an "envelope", and we might hypothesize that this envelope contains the curves, in scaled form, for all M/G/1 queueing systems, not just for the $M/E_k/1$ queues that are studied here. Figure 5.1 shows the upper and lower bounds for the ($\rho = .5$, $z^* = 0$) case. We observe that the envelope is relatively narrow; specifically, its maximum width is 11% of the steady-state value in this case. For $z^* = 0$ and $\rho = .6, .7, .8$, and $.9$, the maximum width is 9%, 7%, 5%, and 3%, respectively. We could use these bounds to obtain approximate results for any M/G/1 queueing system for which λ, $E(S)$, $E(S^2)$, and z are known, without specifying the exact shape of the service time distribution.

Another monotonic ordering is observed by fixing the values of k and z^* and varying ρ. For example, if we examine the curves for the M/D/1 queue ($k = \infty$) with $z^* = 0$, we observe that for $\rho < 1$ the scaled curves increase monotonically as ρ increases. (In Appendix C this comparison requires examining tables that are each three pages apart.) The M/D/1 curves seem to be approaching the curve for the Wiener process as ρ increases. This observation has not been proven analytically, but the heavy traffic (or Brownian motion) approximation for the equilibrium distribution of M/G/1 server load is derived by examining, the scaled form, the limit of a sequence of queues as ρ

ρ \ t^*	.1	.2	.4	.6	1.	2.	4.
2.0	91	60	39	30	21	13	7
1.5	56	38	25	19	14	8	5
1.4	46	32	21	16	12	7	4
1.3	37	25	17	13	10	6	3
1.2	25	18	12	9	7	4	2
1.1	13	9	6	5	4	2	1
.9	14	10	6	4	3	1	0
.8	35	23	14	10	7	3	1
.7	65	42	26	18	12	6	2
.6	108	68	41	30	19	9	3
.5	171	109	67	48	31	15	5

The tabulated error is the difference between the Wiener and the gamma input values, divided by gamma input value, and rounded to the nearest whole per cent.

FIGURE 5.2. Percentage Error of the Wiener Approximation, $z^* = 0$

relationship using a quadratic function. In such cases knowledge of both the first and second moments of the server load distribution is required. Prabhu [1965] derived the double Laplace transform for the time-dependent server load distribution in M/G/1 queues. If we evaluate the second derivative of that expression at zero, we obtain the Laplace transform for the time-dependent second moment of the distribution. The LINV algorithm can then obtain the second moment at specified epochs for use in a quadratic cost function. There may be other queueing applications for this relatively efficient and accurate numerical technique for Laplace transform inversion.

REFERENCES

Bhat, U.N. [1969], "Sixty Years of Queueing Theory," Management Science, Vol. 15, No. 6, pp. B280-B294.

Blumentahal, R.M., and Getoor, R.K. [1968], Markov Processes and Potential Theory, Academic Press, New York.

Breiman, L. [1968], Probability, Addison-Wesley, Reading, Mass.

Coleman, D.R. [1975], "A Monte Carlo Investigation of Stochastic Processes in Single-Server Queues," Ph.D. Dissertation, Stanford University, Stanford, California.

Drake, A.W. [1967], Fundamentals of Applied Probability Theory, McGraw-Hill, New York.

Gaver, D.P., Jr. [1966], "Observing Stochastic Processes and Approximate Transform Inversion," Operations Research, Vol. 15, pp. 444-459.

Gaver, D.P., Jr. [1968], "Diffusion Approximations and Models for Certain Congestion Problems," J. Appl. Prob., Vol. 5, pp. 607-623.

Harrison, J.M. [1977], "The Supremum Distribution of a Levy Process with No Negative Jumps," Adv. Appl. Prob., Vol. 9, pp. 417-422.

Henrici, P. [1964], Elements of Numerical Analysis, Wiley, New York.

Hillier, F.S. and Lieberman, G.J. [1974], Introduction to Operations Research (Second Edition), Holden-Day, San Francisco.

Hora, S.C. [1978], "A Screening Test for the Poisson Process," Decision Sciences, Vol. 9, No. 3, pp. 414-420.

IBM [1971], IBM System/360 and System/370 FORTRAN IV Language, IBM, New York.

Lee, A.H. [1966], Applied Queueing Theory, Macmillan, London.

Pearson, K., (editor) [1922], Tables of the Incomplete Gamma Function, Cambridge University Press.

Prabhu, N.U. [1965], Queues and Inventories, Wiley, New York.

Reinmuth, J.E. [1971], "A Test for the Detection of a Poisson Process," Decision Sciences, Vol. 2, No. 3, pp. 260-263.

Stehfest, H. [1970], "Algorithm 368 -- Numerical Inversion of Laplace Transforms," Comm. of the ACM, Vol. 13, No. 1, pp. 47-49.

Stehfest, H. [1970], "Remark on Algorithm 368," Comm. of the ACM, Vol. 13, No. 10, p. 624.

Takacs, L. [1967], Combinatorial Methods in the Theory of Stochastic Processes, Wiley, New York.

Veillon, F. [1974], "Algorithm 486 -- Numerical Inversion of Laplace Transform," Comm. of the ACM, Vol. 17, No. 10, pp. 587-589.

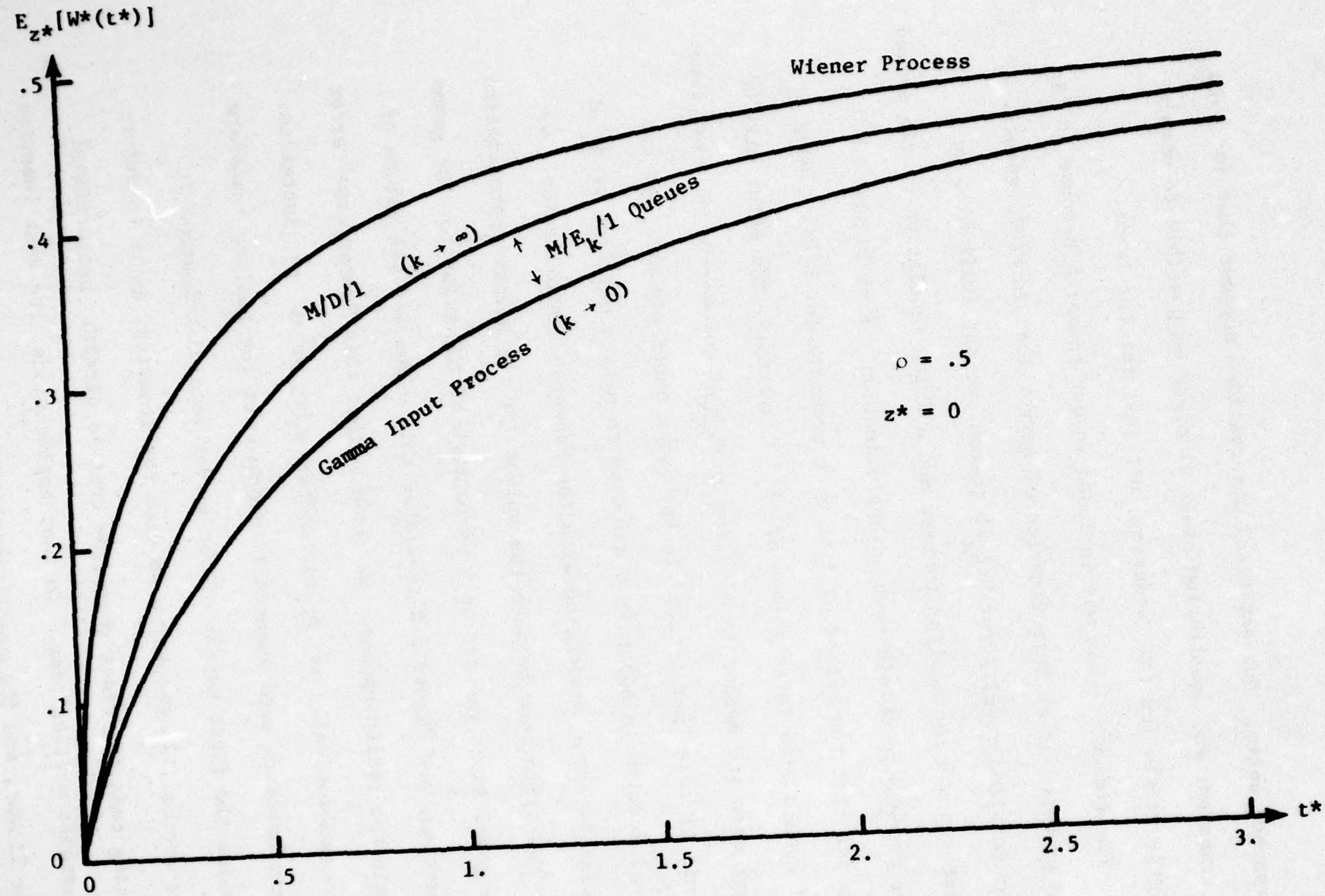


FIGURE 5.1. Several Scaled Processes.

approaches unity. Our empirical observations suggest that for both the transient and equilibrium cases it might be possible to obtain analytic proofs for the ordering, not just for the limit.

The Wiener process was included in our research because it is often proposed as an approximation of server load in M/G/1 queues. If our detailed results for $M/E_k/1$ queues are not available, an analyst can use the scaling factors and a brief tabulation of the scaled Wiener process to obtain such an approximation. From Figure 5.1 we can see that the method will be most accurate for M/D/1 queues or $M/E_k/1$ queues with large values of k. Conversely, the error will be greatest when the method is used for an $M/E_k/1$ queue with k near zero. The limit of this "worst case" is the gamma input process, and our numerical results in Appendix C allow us to make a detailed study of the possible error associated with the Brownian approximation. We express the difference between the values for the Wiener approximation and the gamma input process as a percentage of the value for the gamma input process, and Figure 5.2 tabulates this error for all values of ρ at selected scaled epochs. We observe that this "worst case" error decreases monotonically as ρ approaches unity or as $t*$ increases.

This research used numerical inversion of the Laplace transform to determine the first moment of the server load distribution at specified epochs. In Chapter 4 we used the transient curve to determine waiting cost for cases where the cost is directly proportional to the customer waiting time. In many applications, the cost function may not be linear, and the practitioner may approximate the cost

APPENDIX A

SUMMARY OF NOTATION

This list includes brief definitions of the notation used most frequently in this dissertation.

$A(t)$	stationary Poisson process, the M/G/1 arrival process
c	cost per unit of waiting time
$E(\cdot)$	expectation, first moment, of a random variable
$E_z[\cdot]$	expected value, conditional on initial load z
EF	scaling factor for epochs
Fa	an approximation based on Stehfest's algorithm
$F(\cdot)$	cumulative probability distribution for S
$F^*(s)$	Laplace-Stieltjes transform of $F(\cdot)$
$f(\cdot)$	probability density function
$f_n(t; a)$	observational density function with parameters a and n
$G(t)$	gamma process
$g(t)$	correction term function
$H(t)$	$E_z[I(t)] + g(t)$
$I(t)$	cumulative idleness process
k	Erlang shape parameter
N	number of values of the Laplace transform used by Stehfest's algorithm to obtain an approximation
$P(t)$	any function of interest
$\bar{P}_n$	an approximation of $P(t)$ based on n values of the Laplace transform using Gaver's method

$P_H(z,s)$	Laplace transform of $H(t)$, conditional on z
$P_W(z,s)$	Laplace transforn of $W(t)$, conditional on z
$p(s)$	any Laplace transform function
S, S_i	service time random variable
$S(t)$	compound Poisson process, an input process
T	random variable for an observational epoch
$Var[\cdot]$	variance of a random variable or process
W	random variable for equilibrium server load
WF	scaling factor for waiting time
$W(t)$	server load, or virtual waiting time, process
$X(t)$	net input process, a Levy process
z	deterministic initial server load, $W(0)$
α_i	error components of an asymptotic expansion
α, β	shape and scale parameters for gamma density function
$\Gamma(\cdot)$	gamma function
λ	mean arrival rate, parameter for $A(t)$
μ	$-E[X(t)]/t$
$\xi(t)$	standard Wiener process
ρ	traffic intensity parameter
σ^2	$Var[X(t)]/t$
$\Phi(\cdot)$	exponent function
$\omega(s)$	solution of functional equation $\Phi[\omega(s)] = s$
-	overscore, or "bar", denotes approximation
*	asterisk superscript, denotes a scaled value

APPENDIX B

LISTINGS OF THE COMPUTER PROGRAMS

1. Figure 3.2, including the documented version of LINV
2. Figure 3.4
3. Figure 3.5
4. Figure 3.6
5. Appendix Tables, $\rho \neq 1$
6. Appendix Tables, $\rho = 1$

COMPUTER PROGRAM FOR FIGURE 3.2

```
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      COMMON/LAPLAC/EPOCH(10),APPROX(10,6)
      DIMENSION EXACT(10,6),STEHF(10,6),V(50)
      EXTERNAL P1,P2,P3,P4,P5,P6
      M=0
      N=34
      DATA STEHF /.56555,.39912,.32655,.28278,
     1.25174,.22989,.21322,.19956,.18814,.17796,
     1.16566,1.32543,4.47354,
     110.60342,20.70645,35.78632,56.82535,84.82735,
     1120.78473,165.66749,
     1.98775,.91001,.63826,.30966,
     1-.02119,-.31927,-.57254,-.76869,-.91049,-.98949,
     1-.57762,-1.27084,-1.67544,
     1-1.96392,-2.18727,-2.36870,-2.52270,-2.65740,
     1-2.77390,-2.88091,
     1.36798,.13557,.05043,.01849,
     1.00640,.00195,.00036,-.00006,-.00047,-.00020,
     1-.86533,-.32531,1.02575,2.39533,
     12.78644,1.21092,-3.32956,-11.82953,-25.28353,-44.86511/
      DO 30 I=1,10
      EPOCH(I)=I
      T = I
      EXACT(I,1) =1/DSQRT(T*3.14159265358979323846264 33)
      EXACT(I,2) =T*T*T/6
      EXACT(I,3) =DSIN(DSQRT(2*T))
      EXACT(I,4) =-DLOG(T)-0.5772156649015338 6061
      EXACT(I,5) =DEXP(-T)
      EXACT(I,6) = 1-3*T+3*T*T/2-T*T*T/6
      CALL LINV(P1,N,T,APPROX(I,1),V,M)
      CALL LINV(P2,N,T,APPROX(I,2),V,M)
      CALL LINV(P3,N,T,APPROX(I,3),V,M)
      CALL LINV(P4,N,T,APPROX(I,4),V,M)
      CALL LINV(P5,N,T,APPROX(I,5),V,M)
      CALL LINV(P6,N,T,APPROX(I,6),V,M)
   30 CONTINUE
      DO 80 NCOPY=1,5
      WRITE(6,4)
    4 FORMAT(1H1,////)
      MARGIN = 4 + NCOPY
      DO 8 IMRGN = 5,MARGIN
      WRITE(6,6)
    6 FORMAT(1H )
    8 CONTINUE
      WRITE(6,40)
   40 FORMAT(1H ,35X,'FIGURE 3.2  INVERSION OF TEST',
     1' FUNCTIONS')
      WRITE(6,42)
   42 FORMAT(1H ,35X,39('-'))
      WRITE(6,43)
   43 FORMAT(1H0,33X,2('APPROXIMATE F(T) USING',21X))
```

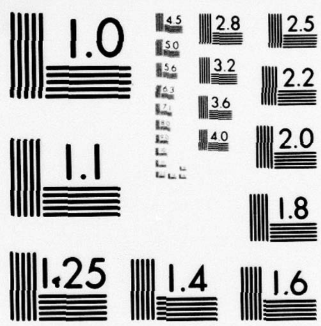
COMPUTER PROGRAM FOR FIGURE 3.2

```
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      COMMON/LAPLAC/EPOCH(10),APPROX(10,6)
      DIMENSION EXACT(10,6),STEHF(10,6),V(50)
      EXTERNAL P1,P2,P3,P4,P5,P6
      M=0
      N=34
      DATA STEHF /.56555,.39912,.32655,.28278,
     1.25174,.22989,.21322,.19956,.18814,.17796,
     1.16566,1.32543,4.47354,
     110.60342,20.70845,35.78632,56.82535,84.82735,
     1120.78473,165.66749,
     1.98775,.91001,.63826,.30966,
     1-.02119,-.31927,-.57254,-.76869,-.91049,-.98949,
     1-.57762,-1.27084,-1.67544,
     1-1.96392,-2.18727,-2.36870,-2.52270,-2.65740,
     1-2.77390,-2.88091,
     1.36798,.13557,.05043,.01849,
     1.00640,.00195,.00036,-.00006,-.00047,-.00020,
     1-.66533,-.32531,1.02575,2.39533,
     12.78844,1.21092,-3.32956,-11.82953,-25.28393,-44.86511/
      DO 30 I=1,10
      EPOCH(I)=I
      T = I
      EXACT(I,1)=1/DSQRT(T*3.14159265358979323846626433)
      EXACT(I,2)=T*T*T/6
      EXACT(I,3)=DSIN(DSQRT(2*T))
      EXACT(I,4)=-DLOG(T)-0.5772156649015338606l
      EXACT(I,5)=DEXP(-T)
      EXACT(I,6)=1-3*T+3*T*T/2-T*T*T/6
      CALL LINV(P1,N,T,APPROX(I,1),V,M)
      CALL LINV(P2,N,T,APPROX(I,2),V,M)
      CALL LINV(P3,N,T,APPROX(I,3),V,M)
      CALL LINV(P4,N,T,APPROX(I,4),V,M)
      CALL LINV(P5,N,T,APPROX(I,5),V,M)
      CALL LINV(P6,N,T,APPROX(I,6),V,M)
   30 CONTINUE
      DO 80 NCOPY=1,5
      WRITE(6,4)
    4 FORMAT(1H1,////)
      MARGIN = 4 + NCOPY
      DO 8 IMRGN = 5,MARGIN
      WRITE(6,6)
    6 FORMAT(1H )
    8 CONTINUE
      WRITE(6,40)
   40 FORMAT(1H ,35X,'FIGURE 3.2  INVERSION OF TEST',
     1' FUNCTIONS')
      WRITE(6,42)
   42 FORMAT(1H ,35X,39('-'))
      WRITE(6,43)
   43 FORMAT(1H0,33X,2('APPROXIMATE F(T) USING',21X))
```

AD-A073 744 STANFORD UNIV CALIF DEPT OF OPERATIONS RESEARCH F/G 12/1
TRANSIENT EFFECTS IN M/G/1 QUEUES: AN EMPIRICAL INVESTIGATION.(U)
JUN 79 M R MIDDLETON N00014-76-C-0418
UNCLASSIFIED TR-85 NL

2 OF 2
ADA
073744

END
DATE
FILMED
10 -79
DDC



MICROCOPY RESOLUTION TEST CHART

```
      WRITE(6,44)
 44   FORMAT(1H ,35X,2(17HSTEHFEST'S METHOD,26X))
      WRITE(6,45)
 45   FORMAT(1H ,32X,2(25('-'),18X))
      WRITE(6,46) N,M
 46   FORMAT(1H ,11X,'T',6X,2('EXACT F(T) ',9X,'N=',I2,9X,
     1'N=10',7X))
      WRITE(6,48)
 48   FORMAT(1H ,10X,'----',2(2X,14('-')),2X,9('-'),
     12(2X,14('-')),2X,9('-'))
      WRITE(6,50)
 50   FORMAT(1H0,26X,'F(T) = 1/SQRT(PI*T)',24X,
     1'F(T) = -C-LN(T)')
      WRITE(6,52)
 52   FORMAT(1H ,16X,2(41('-'),2X))
      DO 56 I=1,10
      WRITE(6,54) EPOCH(I), (EXACT(I,J),APPROX(I,J),
     1STEHF(I,J), J=1,4,3)
 54   FORMAT(1H ,10X,F4.1,2(2(2X,F14.10),2X,F9.5))
 56   CONTINUE
      WRITE(6,60)
 60   FORMAT(1H0,26X,'F(T) = (T*T*T)/6',27X,
     1'F(T) = EXP(-T)')
      WRITE(6,52)
      DO 66 I=1,10
      WRITE(6,54) EPOCH(I), (EXACT(I,J),APPROX(I,J),
     1STEHF(I,J), J=2,5,3)
 66   CONTINUE
      WRITE(6,70)
 70   FORMAT(1H0,26X,'F(T) = SIN(SQRT(2*T))',22X,
     1'F(T) = 1-3*T+3*T*T/2-T*T*T/6')
      WRITE(6,52)
      DO 76 I=1,10
      WRITE(6,54) EPOCH(I), (EXACT(I,J),APPROX(I,J),
     1STEHF(I,J), J=3,6,3)
 76   CONTINUE
 80   CONTINUE
      STOP
      END
C
C
      SUBROUTINE LINV(P,N,T,FA,V,M)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/INPUT/EPOCH(10),LAMBDA,EXPSVC,RHO,JTOL(8)
      COMMON/OUTPUT/QUAL(10),EXPWT(10,8),ITNTOT(8)
      DIMENSION E(25), F(25), H(25), V(50)
C
C..ARGUMENT P = FUNCTION P(S) = LAPLACE TRANSFORM OF F(T)
C          N = NO. OF VALUES OF S USED TO APPROXIMATE F(T)
C          T = VALUE AT WHICH F(T) IS TO BE APPROXIMATED
C          FA= APPROXIMATE F(T) BASED ON N VALUES OF P(S)
C          V = ARRAY OF COEFFICIENTS (USED REPEATEDLY)
C          M = FORMAL PARAMETER EXPLAINED BELOW.
```

```
      WRITE(6,44)
 44   FORMAT(1H ,35X,2(17HSTEHFEST'S METHOD,26X))
      WRITE(6,45)
 45   FORMAT(1H ,32X,2(25('-'),18X))
      WRITE(6,46) N,N
 46   FORMAT(1H ,11X,'T',6X,2('EXACT F(T)',9X,'N=',I2,9X,
     1'N=10',7X))
      WRITE(6,48)
 48   FORMAT(1H ,10X,'----',2(2X,14('-')),2X,9('-'),
     12(2X,14('-')),2X,9('-'))
      WRITE(6,50)
 50   FORMAT(1H0,26X,'F(T) = 1/SQRT(PI*T)',24X,
     1'F(T) = -C-LN(T)')
      WRITE(6,52)
 52   FORMAT(1H ,16X,2(41('-'),2X))
      DO 56 I=1,10
      WRITE(6,54) EPOCH(I), (EXACT(I,J),APPROX(I,J),
     1STEHF(I,J), J=1,4,3)
 54   FORMAT(1H ,10X,F4.1,2(2(2X,F14.10),2X,F9.5))
 56   CONTINUE
      WRITE(6,60)
 60   FORMAT(1H0,26X,'F(T) = (T*T*T)/6',27X,
     1'F(T) = EXP(-T)')
      WRITE(6,52)
      DO 66 I=1,10
      WRITE(6,54) EPOCH(I), (EXACT(I,J),APPROX(I,J),
     1STEHF(I,J), J=2,5,3)
 66   CONTINUE
      WRITE(6,70)
 70   FORMAT(1H0,26X,'F(T) = SIN(SQRT(2*T))',22X,
     1'F(T) = 1-3*T+3*T*T/2-T*T*T/6')
      WRITE(6,52)
      DO 76 I=1,10
      WRITE(6,54) EPOCH(I), (EXACT(I,J),APPROX(I,J),
     1STEHF(I,J), J=3,6,3)
 76   CONTINUE
 80   CONTINUE
      STOP
      END
C
C
      SUBROUTINE LINV(P,N,T,FA,V,M)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/INPUT/EPOCH(10),LAMBDA,EXPSVC,RHO,JTOL(8)
      COMMON/OUTPUT/QUAL(10),EXPWT(10,8),ITNTOT(8)
      DIMENSION E(25), F(25), H(25), V(50)
C
C..ARGUMENT P = FUNCTION P(S) = LAPLACE TRANSFORM OF F(T)
C          N = NO. OF VALUES OF S USED TO APPROXIMATE F(T)
C          T = VALUE AT WHICH F(T) IS TO BE APPROXIMATED
C          FA= APPROXIMATE F(T) BASED ON N VALUES OF P(S)
C          V = ARRAY OF COEFFICIENTS (USED REPEATEDLY)
C          M = FORMAL PARAMETER EXPLAINED BELOW.
```

```
      IF (M.EQ.N) GO TO 50
C
C..ON FIRST CALL OF LINV (M NOT EQUAL TO N) ARRAY V(I) MUST
C  BE EVALUATED.
C
C..FIRST COMPUTE E(I) = FACTORIAL OF 2I OVER THOSE OF I AND
C  I-1 , AND F(I) = FACTORIAL OF I-1.
C
      NH = N/2
      E(1) = 2
      E(2) = 12
      F(1) = 1
      F(2) = 1
C
      DO 10 I = 2, NH
      E(I+1) = ( 2.D0*(2*I+1)*E(I) ) / I
      F(I+1) = F(I)*I
   10 CONTINUE
C
C..NEXT COMPUTE H(J) = ALL TERMS INDEPENDENT OF INDEX I IN
C  THE V(I) SUMMATION.
C
      PWR = NH
      H(1) = 2/F(NH)
C
      DO 20 J = 2, NH
      H(J) = ((J**PWR)*E(J))/F(NH-J+1)
   20 CONTINUE
C
C..FINALLY EVALUATE ARRAY V(I).
C
      ISIGN = 2*(NH-2*(NH/2))-1
C
      DO 40 I = 1, N
      V(I) = 0
      JFIRST = (I+1)/2
      JLAST = MIN0(I,NH)
C
      DO 30 J = JFIRST, JLAST
      V(I) = V(I) + H(J)/(F(I-J+1)*F(2*J-I+1))
   30 CONTINUE
C
      V(I) = ISIGN*V(I)
      ISIGN = -ISIGN
   40 CONTINUE
C
      M = N
C
C..SUBSEQUENT CALLS (M EQUAL TO N) WILL USE ARRAY V(I) FROM
C  COMMON STORAGE AND WILL BRANCH TO THIS SECTION OF LINV.
C
   50 FA = 0
      A = .693147180559945309417232121458D0 / T
```

```
      DO 60 I= 1, N
      FA = FA + V(I)*P(I*A)
   60 CONTINUE
C
      FA = A*FA
C
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION P1(S)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      P1=1/DSQRT(S)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION P2(S)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      P2=1/(S*S*S*S)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION P3(S)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      P3=DSQRT(3.14159265358979323846264433/(2*S*S*S))
     1*DEXP(-1/(2*S))
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION P4(S)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      P4=DLOG(S)/S
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION P5(S)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      P5=1/(S+1)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION P6(S)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      P6=(S-1)*(S-1)*(S-1)/(S*S*S*S)
      RETURN
      END
```

COMPUTER PROGRAM FOR FIGURE 3.4

```
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      COMMON FA1(10),FA2(9)
      DIMENSION F1(10),S1(10),D1(10),V1(10),
     1F2(9),B2(9),V2(9),V(50),T2(9)
      EXTERNAL F1,F2
      M=0
      N=34
      DO 20 I=1,10
      T=I
      F1(I)=1/DSQRT(T*3.1415926535897932384626433)
      CALL LINV(F1,N,T,FA1(I),V,M)
   20 CONTINUE
C
      DATA S1/.56555,.39912,.32655,.28276,
     1.25174,.22989,.21322,.19956,.18814,.17796/
      DATA D1/.73172,.40035,.26343,.28286,
     1.29365,.22901,.18062,.20112,.21609,.17650/
      DATA V1/.56419,.39894,.32573,.28209,
     1.25231,.23033,.21324,.19947,.18806,.17841/
      DATA T2/4.140186,2.501126,1.643438,1.085064,
     1.693147,.412298,.214821,.085541,.016048/
C
      DO 30 I=1,9
      T=T2(I)
      F2(I)=DEXP(-T/2)
      CALL LINV(F2,N,T,FA2(I),V,M)
   30 CONTINUE
C
      DATA B2/.120527,.288195,.439084,.581306,
     1.707316,.813401,.898482,.957847,.992205/
      DATA V2/.126174,.286329,.439675,.581269,
     1.707107,.813712,.898156,.958135,.992015/
C
      DO 180 NCOPY=1,5
      WRITE(6,4)
    4 FORMAT(1H1,////)
      MARGIN = 4 + NCOPY
      DO 8 IMAGN = 5,MARGIN
      WRITE(6,6)
    6 FORMAT(1H )
    8 CONTINUE
C
      WRITE(6,40)
   40 FORMAT (1H ,33X,'FIGURE 3.4  COMPARISONS',
     1' WITH OTHER TECHNIQUES')
      WRITE(6,48)
   48 FORMAT(1H ,33X,45('-'))
      WRITE(6,52)
   52 FORMAT(1H0,20X,'F(T) = 1/SQRT(T*PI)')
```

```
      WRITE(6,54)
   54 FORMAT(1H ,20X,19('-'),3X,'APPROXIMATE F(T)',
     1' USING NUMERICAL INVERSION')
      WRITE(6,60)
   60 FORMAT(1H ,42X,42('-'))
      WRITE(6,64)
   64 FORMAT(1H ,44X,17HSTEHFEST'S METHOD)
      WRITE(6,66)
   66 FORMAT(1H ,42X,21('-'))
      WRITE(6,70) N
   70 FORMAT(1H ,22X,'T       EXACT F(T)        N=',I2,6X,
     1'N=10*      DUBNER*     VEILLON*')
      WRITE(6,72)
   72 FORMAT(1H ,21X,'---',2(4X,10('-')),3(4X,7('-')))
      DO 78 I=1,10
      WRITE(6,76) I,F1(I),FA1(I),S1(I),D1(I),V1(I)
   76 FORMAT(1H ,21X,I2,5X,2(F10.8,4X),3(F7.5,4X))
   78 CONTINUE
C
      WRITE(6,80)
   80 FORMAT(1H-,20X,'F(T) = EXP(-T/2)')
      WRITE(6,82)
   82 FORMAT(1H ,20X,16('-'),10X,'APPROXIMATE F(T)',
     1' USING NUMERICAL INVERSION')
      WRITE(6,85)
   85 FORMAT(1H ,46X,42('-'))
      WRITE(6,86) N
   86 FORMAT(1H ,24X,'T',9X,'EXACT F(T)',6X,
     1'STEHFEST N=',I2,4X,'BELLMAN*    VEILLON*')
      WRITE(6,88)
   88 FORMAT(1H ,20X,8('-'),2(3X,15('-')),2(3X,8('-')))
      DO 94 I=1,9
      WRITE(6,92) T2(I),F2(I),FA2(I),B2(I),V2(I)
   92 FORMAT(1H ,20X,F8.6,2(3X,F15.13),2(3X,F8.6))
   94 CONTINUE
C
      WRITE(6,102)
  102 FORMAT(1H-,25X,'*NOTE: DATA FOR METHODS MARKED',
     1' BY AN ASTERISK WERE TAKEN')
      WRITE(6,104)
  104 FORMAT(1H ,32X,'FROM A.C.M ALGORITHM 486, ',
     123H'NUMERICAL INVERSION OF)
      WRITE(6,106)
  106 FORMAT(1H ,32X,23HLAPLACE TRANSFORM', BY ,
     1'FRANCOISE VEILLON, COMMUNI-')
      WRITE(6,108)
  108 FORMAT(1H ,32X,'CATIONS OF THE A.C.M.,',
     1' VOLUME 17, NUMBER 10,')
      WRITE(6,110)
  110 FORMAT(1H ,32X,'OCTOBER 1974.')
  180 CONTINUE
      STOP
      END
```

```
      SUBROUTINE LINV(P,N,T,FA,V,M)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER (I-N)
      COMMON FA1(10),FA2(9)
      DIMENSION E(25), F(25), H(25), V(50)
      IF (M.EQ.N) GO TO 50
      NH = N/2
      E(1) = 2
      E(2) = 12
      F(1) = 1
      F(2) = 1
      DO 10 I = 2, NH
      E(I+1) = ( 2.D0* (2*I+1)*E(I) ) / I
      F(I+1) = F(I)*I
10    CONTINUE
      PNH = NH
      H(1) = 2/F(NH)
      DO 20 J = 2, NH
      H(J) = ((J**PNH)*E(J))/F(NH-J+1)
20    CONTINUE
      ISIGN = 2*(NH-2*(NH/2))-1
      DO 40 I = 1, N
      V(I) = 0
      JFIRST = (I+1)/2
      JLAST = MIN0(I,NH)
      DO 30 J = JFIRST, JLAST
      V(I) = V(I) + H(J)/(F(I-J+1)*F(2*J-I+1))
30    CONTINUE
      V(I) = ISIGN*V(I)
      ISIGN = -ISIGN
40    CONTINUE
      M = N
50    FA = 0
      A = .69314718055994530941723212145D0 / T
      DO 60 I= 1, N
      FA = FA + V(I)*P(I*A)
60    CONTINUE
      FA = A*FA
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION P1(S)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      P1=1/DSQRT(S)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION P2(S)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      P2=2/(1+2*S)
      RETURN
      END
```

COMPUTER PROGRAM FOR FIGURE 3.5

```
      IMPLICIT REAL*8(A-H,K,L,O-Z) , INTEGER(I,J,M,N)
      COMMON/INPUT/EPOCH(10),LAMBDA,EXPSVC,RHO,JTOL(8)
      COMMON/OUTPUT/QUAD(10),EXPMT(10,8),ITNTOT(8)
      COMMON/LAPLAC/V(50),M,N
      COMMON/MM1/OSTART,SS
      COMMON/NEWRAP/TOLRNC,ITNSUM
      EXTERNAL FQUAD,PMM1
      EXTERNAL FMM1,DFMM1,DDFMM1
      M = 0
      N = 34
C     Z        = 0.0C
C
      DO 20 I = 1,10
      CALL LINV(FQUAD,N,EPOCH(I),EZIT,V,M)
      QUAD(I) = (RHO-1.)*EPOCH(I) + EZIT
   20 CONTINUE
C
      DO 50 J=1,8
      JTOL(J) = 2*J + 8
      TOLRNC = 1./(10.**JTOL(J))
      OSTART = 1
      ITNSUM = 0
C
      DO 40 I=1,10
      CALL LINV(PMM1,N,EPOCH(I),EZIT,V,M)
      EXPMT(I,J) = (RHO-1.)*EPOCH(I) + EZIT
   40 CONTINUE
C
      ITNTOT(J) = ITNSUM
C
   50 CONTINUE
C
      CALL TOLOUT
C
      STOP
      END
C
C
      BLOCK DATA
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/INPUT/EPOCH(10),LAMBDA,EXPSVC,RHO,JTOL(8)
      DATA EPOCH /20.,40.,60.,80.,100.,
     1200.,300.,400.,800.,1200./
      DATA LAMBDA,EXPSVC,RHO /1.00,0.95,0.95/
      END
C
C
      SUBROUTINE TOLOUT
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/INPUT/EPOCH(10),LAMBDA,EXPSVC,RHO,JTOL(8)
      COMMON/OUTPUT/QUAD(10),EXPMT(10,8),ITNTOT(8)
```

```
      COMMON/LAPLAC/V(50),N,N
      DIMENSION BESSEL(5),JTOL(8)
C
      BESSEL(1) = 3.9181
      BESSEL(2) = 5.4651
      BESSEL(3) = 6.5582
      BESSEL(4) = 7.4191
      BESSEL(5) = 8.1335
C
      DO 80 NCOPY=1,5
      WRITE(6,4)
    4 FORMAT(1H1,////)
      MARGIN = 4 + NCOPY
      DO 8 IMRGN = 5,MARGIN
      WRITE(6,6)
    6 FORMAT(1H )
    8 CONTINUE
C
      WRITE(6,12)
   12 FORMAT(1H ,29X,'FIGURE 3.5  EFFECT OF NEWTON-RAPHSON',
     1' SEARCH TOLERANCE')
C
      WRITE(6,14)
   14 FORMAT(1H ,29X,53('-'))
C
      WRITE(6,16)
   16 FORMAT(1H0,10X,37HTHIS TABLE COMPARES COLEMAN'S BESSEL ,
     1'FUNCTION RESULTS AT FIVE EPOCHS WITH LAPLACE')
C
      WRITE(6,18)
   18 FORMAT(1H ,10X,'INVERSION RESULTS FOR BOTH THE EXACT ',
     1'QUADRATIC SOLUTION OF THE FUNCTIONAL AND THE ')
C
      WRITE(6,20)
   20 FORMAT(1H ,10X,'APPROXIMATE NEWTON-RAPHSON SEARCH ',
     1'SOLUTION OF THE FUNCTIONAL WITH VARIOUS TOLERANCES.')
      WRITE(6,30)
   30 FORMAT(1H0,10X,'M/M/1 MEAN SERVER LOAD,',
     1' LAMBDA = 1.0, E(S) = 0.95, Z = 0.')
C
      DO 80 JFIRST=1,5,4
      JLAST=JFIRST+3
C
      WRITE(6,38) N
   38 FORMAT(1H0,36X,'INVERSION OF LAPLACE TRANSFORM,',
     122H STEHFEST'S METHOD, N=,I2)
C
      WRITE(6,40)
   40 FORMAT(1H ,18X,'SUMS OF  ',73('-'))
C
      WRITE(6,42)
   42 FORMAT(1H ,19X,'BESSEL    QUADRATIC    TOLERANCE FOR ',
     1'NEWTON-RAPHSON SEARCH SOLUTION OF FUNCTIONAL')
```

```
C
      WRITE(6,44)
   44 FORMAT(1H ,10X,'EPOCH  FUNCTIONS  SOLUTION OF   ',58('-'))
C
      DO 46 J=JFIRST,JLAST
   46 CONTINUE
C
      WRITE(6,50) (JTOL(J),J=JFIRST,JLAST)
   50 FORMAT(1H ,10X,' T    (COLEMAN)   FUNCTIONAL     ',
     14('10**-',I2,8X))
C
      WRITE(6,56)
   56 FORMAT(1H ,10X,'-----       ------',5(2X,13('-')))
C
      DO 62 I=1,5
C
      WRITE(6,60) EPOCH(I),BESSEL(I),QUAD(I),
     1(EXPNT(I,J),J=JFIRST,JLAST)
   60 FORMAT(1H ,10X,F5.0,3X,F7.4,5(1X,F14.11))
   62 CONTINUE
C
      DO 66 I=6,10
C
      WRITE(6,64) EPOCH(I),QUAD(I),(EXPNT(I,J),J=JFIRST,JLAST)
   64 FORMAT(1H ,10X,F5.0,10X,5(1X,F14.11))
C
   66 CONTINUE
C
      WRITE(6,70)
   70 FORMAT(1H ,10X,'TOTAL NUMBER OF NEWTON-RAPHSON')
C
      WRITE(6,72) (ITNTOT(J),J=JFIRST,JLAST)
   72 FORMAT(1H ,10X,'ITERATIONS TO EVALUATE 10 EPOCHS:    ',
     14(I4,11X))
C
   80 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE LINV(P,N,T,FA,V,M)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      COMMON/LAPLAC/EPOCH(10),APPROX(10,6)
      DIMENSION E(25), F(25), H(25), V(50)
      IF (M.EQ.N) GO TO 50
      NH = N/2
      E(1) = 2
      E(2) = 12
      F(1) = 1
      F(2) = 1
      DO 10 I = 2, NH
      E(I+1) = ( 2.D0*(2*I+1)*E(I) ) / I
      F(I+1) = F(I)*I
```

```
10    CONTINUE
      FNH = NH
      H(1) = 2/F(NH)
      DO 20 J = 2, NH
      H(J) = ((J**FNH)*E(J))/F(NH-J+1)
20    CONTINUE
      ISIGN = 2*(NH-2*(NH/2))-1
      DO 40 I = 1, N
      V(I) = 0
      JFIRST = (I+1)/2
      JLAST = MIN0(I,NH)
      DO 30 J = JFIRST, JLAST
      V(I) = V(I) + H(J)/(F(I-J+1)*F(2*J-I+1))
30    CONTINUE
      V(I) = ISIGN*V(I)
      ISIGN = -ISIGN
40    CONTINUE
      M = N
50    FA = 0
      A = .6931471805599453094172321214580D0 / T
      DO 60 I= 1, N
      FA = FA + V(I)*P(I*A)
60    CONTINUE
      FA = A*FA
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION PQUAD(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/INPUT/EPOCH(10),LAMBDA,EXPSVC,RHO,JTOL(8)
      B = 1 - EXPSVC*(LAMBDA+S)
      OMEGA = (-B+DSQRT(B*B+4*S*EXPSVC))/(2*EXPSVC)
      PQUAD=DEXP(-OMEGA*Z)/(S*OMEGA)
      RETURN
      END
C
      DOUBLE PRECISION FUNCTION PMM1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/INPUT/EPOCH(10),LAMBDA,EXPSVC,RHO,JTOL(8)
      COMMON/MM1/OSTART,SS
      EXTERNAL FOMM1,DFMM1
      SS = S
      CALL ZERO(FOMM1,DFMM1,OSTART,OMEGA)
      OSTART = OMEGA
      PMM1=DEXP(-OMEGA*Z)/(S*OMEGA)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION FMM1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/INPUT/EPOCH(10),LAMBDA,EXPSVC,RHO,JTOL(8)
```

```
      FMM1 = S - LAMBDA*(1-(1/(1+S*EXPSVC)))
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION DFMM1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/INPUT/EPOCH(10),LAMBDA,EXPSVC,RHO,JTOL(8)
      DFMM1 = 1 - LAMBDA*EXPSVC*(1/(1+S*EXPSVC))**2
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION DDFMM1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/INPUT/EPOCH(10),LAMBDA,EXPSVC,RHO,JTOL(8)
      DDFMM1 = 2*LAMBDA*EXPSVC**2*(1/(1+S*EXPSVC))**3
      RETURN
      END
C
      DOUBLE PRECISION FUNCTION FOMM1(OMEGA)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/MM1/OSTART,SS
      FOMM1 = FMM1(OMEGA) - SS
      RETURN
      END
C
C
      SUBROUTINE ZERO(F,DF,XSTART,XZERO)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON/NEWRAP/TOLRNC,ITNSUM
      ITERAT = 0
      X = XSTART
   10 CONTINUE
      XNEW = X - F(X)/DF(X)
      RELACC = (XNEW-X)/XNEW
      X = XNEW
      ITERAT = ITERAT + 1
      IF (ITERAT .GE. 50) GO TO 50
      IF (DABS(RELACC) .LE. TOLRNC) GO TO 20
      GO TO 10
   20 CONTINUE
      ITNSUM = ITNSUM + ITERAT
      XZERO = X
      RETURN
C
   50 CONTINUE
      WRITE(6,55) RELACC, TOLRNC
   55 FORMAT(1H ,'ITERAT=50, RELACC=',D24.16,', TOLRNC=',D24.16)
      ITNSUM = ITNSUM + ITERAT
      XZERO = X
      RETURN
      END
```

COMPUTER PROGRAM FOR FIGURE 3.6

```
COMPUTER PROGRAM FOR FIGURE 3.6
-------------------------------
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      COMMON /RHOZ/ RHO,Z
      DIMENSION SEZIT1(11,8),SEZIT2(11,8),V(50),ZSIN(8),RHOIN(11)
      EXTERNAL PQUAD1,PQUAD2
      DATA ZSIN /.2,.4,.6,.8,1.,2.,3.,4./
      DATA RHOIN /.5,.6,.7,.8,.9,1.1,1.2,1.3,1.4,1.5,2./
C
      M=0
      N=34
C
      DO 12 IRHO=1,11
      RHO = RHOIN(IRHO)
      WTFCTR = DABS(1.-RHO)/RHO
      DO 12 JZ=1,8
      Z = ZSIN(JZ)/WTFCTR
      CALL LINV(PQUAD1,N,Z,EZIZ,V,M)
      SEZIT1(IRHO,JZ) = EZIZ*WTFCTR
      CALL LINV(PQUAD2,N,Z,EZIZ,V,M)
      SEZIT2(IRHO,JZ) = EZIZ*WTFCTR
   12 CONTINUE
C
      DO 88 NCOPY=1,5
      WRITE(6,20)
   20 FORMAT(1H1,////)
      MARGIN = 4 + NCOPY
      DO 28 IMRGN = 5,MARGIN
      WRITE(6,24)
   24 FORMAT(1H )
   28 CONTINUE
C
      WRITE(6,32)
   32 FORMAT(1H ,26X,'FIGURE 3.6  CORRECTIONS',
     1' FOR DISCONTINUOUS FIRST DERIVATIVE')
      WRITE(6,36)
   36 FORMAT(1H ,26X,56('-'))
      WRITE(6,40) N
   40 FORMAT(1H-,33X,'APPROXIMATIONS USING',
     122H STEHFEST'S METHOD, N=,I2)
      WRITE(6,44)
   44 FORMAT(1H-,20X,'E/M/1 SCALED MEAN CUMULATIVE',
     141H IDLENESS AT T'=ABS(MU)*Z' (I.E., AT T=Z))
      WRITE(6,48)
   48 FORMAT(1H0,47X,'WITHOUT CORRECTION TERM')
      WRITE(6,52)
   52 FORMAT(1H0,45X,'SCALED INITIAL SERVER LOAD')
      WRITE(6,56)
   56 FORMAT(1H ,19X,78('-'))
      WRITE(6,60) (ZSIN(JZ), JZ=1,8)
   60 FORMAT(1H ,15X,'RHO',5X,8(F3.1,7X))
      WRITE(6,64)
   64 FORMAT(1H ,15X,'---',8(2X,8('-')))
```

```
      DO 72 IRHO=1,11
      WRITE(6,68) RHOIN(IRHO), (SEZIT1(IRHO,JZ), JZ=1,8)
   68 FORMAT(1H ,15X,F3.1,8(2X,F8.6))
   72 CONTINUE
C
      WRITE(6,76)
   76 FORMAT(1H )
      WRITE(6,44)
      WRITE(6,80)
   80 FORMAT(1H0,48X,'WITH CORRECTION TERM')
      WRITE(6,52)
      WRITE(6,56)
      WRITE(6,60) (ZSIN(JZ), JZ=1,8)
      WRITE(6,64)
      DO 84 IRHO=1,11
      WRITE(6,68) RHOIN(IRHO), (SEZIT2(IRHO,JZ), JZ=1,8)
   84 CONTINUE
   88 CONTINUE
      STOP
      END
C
C
      SUBROUTINE LINV(P,N,T,FA,V,M)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      COMMON/LAPLAC/EPOCH(10),APPROX(10,6)
      DIMENSION E(25), F(25), H(25), V(50)
      IF (M.EQ.N) GO TO 50
      NH = N/2
      E(1) = 2
      E(2) = 12
      F(1) = 1
      F(2) = 1
      DO 10 I = 2, NH
      E(I+1) = ( 2.D0*(2*I+1)*E(I) ) / I
      F(I+1) = F(I)*I
   10 CONTINUE
      PWR = NH
      H(1) = 2/F(NH)
      DO 20 J = 2, NH
      H(J) = ((J**PWR)*E(J))/F(NH-J+1)
   20 CONTINUE
      ISIGN = 2*(NH-2*(NH/2))-1
      DO 40 I = 1, N
      V(I) = 0
      JFIRST = (I+1)/2
      JLAST = MIN0(I,NH)
      DO 30 J = JFIRST, JLAST
      V(I) = V(I) + H(J)/(F(I-J+1)*F(2*J-I+1))
   30 CONTINUE
      V(I) = ISIGN*V(I)
      ISIGN = -ISIGN
   40 CONTINUE
      M = N
```

```
50   FA = 0
     A = .693147180559945309417232121458D0 / T
     DO 60 I= 1, N
     FA = FA + V(I)*F(I*A)
60   CONTINUE
     FA = A*FA
     RETURN
     END
C
C
     DOUBLE PRECISION FUNCTION PQUAD1(S)
     IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
     COMMON /RHOZ/ RHO,Z
     B = 1 - RHO - S/2.
     OMEGA = -B+DSQRT(B*B+2*S)
     PQUAD1 = DEXP(-OMEGA*Z)/(S*OMEGA)
     RETURN
     END
C
C
     DOUBLE PRECISION FUNCTION PQUAD2(S)
     IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
     COMMON /RHOZ/ RHO,Z
     B = 1 - RHO - S/2.
     OMEGA = -B+DSQRT(B*B+2*S)
     PQUAD2=DEXP(-OMEGA*Z)/(S*OMEGA)-DEXP(-Z*(2.*RHO+S))/(S*S)
     RETURN
     END
```

COMPUTER PROGRAM FOR APPENDIX TABLES, RHO NOT EQUAL TO ONE

```
      IMPLICIT REAL*8(A-H,K,L,O-Z) , INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),SCLWT(10,16,3),RHOIN,
     1ZSIN(3)
      COMMON /LAPLAC/ V(50),M,N
      COMMON /NEWRAP/ TOLRNC,MAXITN
C
      M=0
      N=34
      MAXITN=20
      TOLRNC=1.D-20
C
      DO 60 IPAGE=1,3
      READ(5,20) RHOIN, (ZSIN(ITABLE) , ITABLE=1,3)
      READ(5,20)  (TVAL(J), J=1,6)
      READ(5,20)  (TVAL(J), J=7,12)
      READ(5,20)  (TVAL(J), J=13,16)
   20 FORMAT(6F10.0)
C
      DO 50 ITABLE=1,3
      CALL RZWNR(ITABLE)
      CALL RZMD1(ITABLE)
      CALL RZMEK(ITABLE)
      CALL RZGAM(ITABLE)
   50 CONTINUE
      CALL RZOUT
   60 CONTINUE
      STOP
      END
C
C
      BLOCK DATA
      IMPLICIT REAL*8(A-H,K,L,O-Z) , INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),SCLWT(10,16,3),RHOIN,
     1ZSIN(3)
      DATA KVAL/4.,3.,2.,1.5,1.,.5,.2/
      END
C
C
      SUBROUTINE LINV(P,N,T,FA,V,M)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      COMMON/LAPLAC/EPOCH(10),APPROX(10,6)
      DIMENSION E(25), F(25), H(25), V(50)
      IF (M.EQ.N) GO TO 50
      NH = N/2
      E(1) = 2
      E(2) = 12
      F(1) = 1
      F(2) = 1
      DO 10 I = 2, NH
      E(I+1) = ( 2.D0*(2*I+1)*E(I) ) / I
      F(I+1) = F(I)*I
```

```
10    CONTINUE
      PWR = NH
      H(1)  = 2/F(NH)
      DO 20 J = 2, NH
      H(J)  =  ((J**PWR)*E(J))/F(NH-J+1)
20    CONTINUE
      ISIGN = 2*(NH-2*(NH/2))-1
      DO 40 I = 1, N
      V(I)  = 0
      JFIRST =  (I+1)/2
      JLAST = MIN0(I,NH)
      DO 30 J = JFIRST, JLAST
      V(I)  =  V(I)  + H(J)/(F(I-J+1)*F(2*J-I+1))
30    CONTINUE
      V(I)  = ISIGN*V(I)
      ISIGN = -ISIGN
40    CONTINUE
      M = N
50    FA = 0
      A = .693147180559945309417232121458D0 / T
      DO 60 I= 1, N
      FA = FA + V(I)*P(I*A)
60    CONTINUE
      FA = A*FA
      RETURN
      END
C
C
      SUBROUTINE RZWNR(ITABLE)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),SCLWT(10,16,3),RHOIN,
     1ZSIN(3)
      COMMON /LAPLAC/ V(50),M,N
      COMMON /RHOZO/ RHO,Z,OSTART
      EXTERNAL PWNR
C
C     SCALING WITH SIGSQR = 1
      RHO = RHOIN
      ALFSQR =  (1-RHO)*(1-RHO)
      WTFCTR = DABS(1-RHO)
      Z        = ZSIN(ITABLE)/WTFCTR
C
      DO 20 J=1,16
      T = TVAL(J)/ALFSQR
      CALL LINV(PWNR,N,T,TERM3,V,M)
      RZWT = Z +  (RHO-1.)*T + TERM3
      SCLWT(1,J,ITABLE)  = RZWT*WTFCTR
20    CONTINUE
C
      RETURN
      END
```

```
C
C
      DOUBLE PRECISION FUNCTION PWNR(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZO/ RHO,Z,OSTART
      OMEGA = -(1-RHO)+DSQRT((1-RHO)*(1-RHO)+2*S)
      PWNR = DEXP(-OMEGA*Z)/(S*OMEGA)
      RETURN
      END
C
C
      SUBROUTINE RZMD1(ITABLE)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),SCLWT(10,16,3),RHOIN,
     1ZSIN(3)
      COMMON /LAPLAC/ V(50),M,N
      COMMON /RHOZO/ RHO,Z,OSTART
      EXTERNAL PMD1,FMD1,DFMD1,DDFMD1
C
C     SCALING WITH SIGSQR = RHO
      RHO = RHOIN
      ALFSQR = (1-RHO)*(1-RHO)/RHO
      WTFCTR = DABS(1-RHO)/RHO
      OSTART = 1
      Z      = ZSIN(ITABLE)/WTFCTR
      ELZ = DEXP(-RHO*Z)
C
      IF (RHO .LE. 1.0) GO TO 10
      CALL ZERO(DFMD1,DDFMD1,1D-10,SBAR)
      DELTA = -FMD1(SBAR)
      START = SBAR + DELTA
      CALL ZERO(FMD1,DFMD1,START,OSTART)
   10 CONTINUE
      DO 20 J=1,16
      T = TVAL(J)/ALFSQR
      IF (T .LE. Z) EZIT = 0.
      IF (T .GT. Z) CALL LINV(PMD1,M,T,HT,V,N)
      IF (T .GT. Z) EZIT = HT + (T-Z)*ELZ
      EZWT = Z + (RHO-1.)*T + EZIT
      SCLWT(2,J,ITABLE) = EZWT*WTFCTR
   20 CONTINUE
      RETURN
      END
C
C
```

```
      DOUBLE PRECISION FUNCTION FMD1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZO/ RHO,Z,OSTART
      COMMON SS
      EXTERNAL FOMD1,DFMD1
      SS= S
      CALL ZERO(FOMD1,DFMD1,OSTART,OMEGA)
      OSTART = OMEGA
      FMD1 = DEXP(-OMEGA*Z)/(S*OMEGA) - DEXP(-Z*(RHO+S))/(S*S)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION FMD1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZO/ RHO,Z,OSTART
      FMD1 = S-RHO*(1-DEXP(-S))
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION DFMD1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZO/ RHO,Z,OSTART
      DFMD1 = 1-RHO*DEXP(-S)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION DDFMD1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZO/ RHO,Z,OSTART
      DDFMD1 = RHO*DEXP(-S)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION FOMD1(OMEGA)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON SS
      FOMD1 = FMD1(OMEGA) - SS
      RETURN
      END
C
```

```
C
      SUBROUTINE RZMEK(ITABLE)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),SCLWT(10,16,3),RHOIN,
     1ZSIN(3)
      COMMON /LAPLAC/ V(50),M,N
      COMMON /RHOZOK/ RHO,Z,OSTART,K
      EXTERNAL PMEK,FMEK,DFMEK,DDFMEK
C
C     SCALING WITH SIGSQR = (K+1)*RHO/K
      RHO = RHOIN
C
      DO 30 I=3,9
      K        = KVAL(I-2)
      ALFSQR = K*(1-RHO)*(1-RHO)/((K+1)*RHO)
      WTFCTR = K*DABS(1-RHO)/((K+1)*RHO)
      OSTART = 1
      Z        = ZSIN(ITABLE)/WTFCTR
      ELZ = DEXP(-RHO*Z)
C
      IF (RHO .LE. 1.0) GO TO 10
      CALL ZERO(DFMEK,DDFMEK,0,SBAR)
      DELTA = -FMEK(SBAR)
      START = SBAR + DELTA
      CALL ZERO(FMEK,DFMEK,START,OSTART)
   10 CONTINUE
C
      DO 20 J=1,16
      T = TVAL(J)/ALFSQR
      IF (T .LE. Z) EZIT = 0.
      IF (T .GT. Z) CALL LINV(PMEK,M,T,HT,V,M)
      IF (T .GT. Z) EZIT = HT + (T-Z)*ELZ
      EZWT = Z + (RHO-1.)*T + EZIT
      SCLWT(I,J,ITABLE) = EZWT*WTFCTR
   20 CONTINUE
   30 CONTINUE
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION PMEK(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZOK/ RHO,Z,OSTART,K
      COMMON SS
      EXTERNAL FOMEK,DFMEK
      SS= S
      CALL ZERO(FOMEK,DFMEK,OSTART,OMEGA)
      OSTART = OMEGA
      PMEK = DEXP(-OMEGA*Z)/(S*OMEGA) - DEXP(-Z*(RHO+S))/(S*S)
      RETURN
      END
C
```

```
C
      DOUBLE PRECISION FUNCTION FMEK(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZOK/ RHO,Z,OSTART,K
      FMEK = S-RHO*(1-(K/(K+S)))**K)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION DFMEK(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZOK/ RHO,Z,OSTART,K
      DFMEK = 1-RHO*(K/(K+S))**(K+1)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION DDFMEK(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZOK/ RHO,Z,OSTART,K
      DDFMEK = RHO*(K+1)*(K/(K+S))**(K+2)/K
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION FOMEK(OMEGA)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON SS
      FOMEK = FMEK(OMEGA) - SS
      RETURN
      END
C
C
      SUBROUTINE RZGAM(ITABLE)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),SCLMT(10,16,3),RHOIN,
     1ZSIN(3)
      COMMON /LAPLAC/ V(50),M,N
      COMMON /RHOZO/ RHO,Z,OSTART
      EXTERNAL RGAM,FGAM,DFGAM
C
C     SCALING WITH SIGSQR = RHO
      RHO = RHOIN
      ALFSQR = (1-RHO)*(1-RHO)/RHO
      WTFCTR = DABS(1-RHO)/RHO
      OSTART = 1
      Z      = ZSIN(ITABLE)/WTFCTR
C
```

```
      IF (RHO .LE. 1.0) GO TO 10
      SBAR = RHO - 1
      DELTA = -FGAM(SBAR)
      START = SBAR + DELTA
      CALL ZERO(FGAM,DFGAM,START,OSTART)
   10 CONTINUE
      DO 20 J=1,16
      T = TVAL(J)/ALFSQR
      IF (T .LE. Z) EZIT = 0.
      IF (T .GT. Z) CALL LINV(PGAM,N,T,EZIT,V,M)
      EZWT = Z + (RHO-1.)*T + EZIT
      SCLWT(10,J,ITABLE) = EZWT*WTFCTR
   20 CONTINUE
C
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION PGAM(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZO/ RHO,Z,OSTART
      COMMON SS
      EXTERNAL FOGAM,DFGAM
      SS= S
      CALL ZERO(FOGAM,DFGAM,OSTART,OMEGA)
      OSTART = OMEGA
      PGAM = DEXP(-OMEGA*Z)/(S*OMEGA)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION FGAM(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZO/ RHO,Z,OSTART
      FGAM = S-RHO*DLOG(1+S)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION DFGAM(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZO/ RHO,Z,OSTART
      DFGAM = 1-RHO/(1+S)
      RETURN
      END
C
C
```

```
      DOUBLE PRECISION FUNCTION FOGAM(OMEGA)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /RHOZO/ RHO,Z,OSTART
      COMMON SS
      FOGAM = FGAM(OMEGA) - SS
      RETURN
      END
C
C
      SUBROUTINE ZERO(F,DF,XSTART,XZERO)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /NEWRAP/ TOLRNC,MAXITN
      ITERAT = 0
C
      X = XSTART
   10 CONTINUE
      XNEW = X - F(X)/DF(X)
      RELACC = (XNEW-X)/XNEW
      X = XNEW
      ITERAT = ITERAT + 1
      IF (ITERAT .GE. MAXITN) GO TO 50
      IF (DABS(RELACC) .LE. TOLRNC) GO TO 20
      GO TO 10
   20 CONTINUE
      XZERO = X
      RETURN
C
   50 WRITE (6,60)
   60 FORMAT (1H1,'NEWTON-RAPHSON SEARCH IN SUBROUTINE ZERO',
     1' EXCEEDED MAXIMUM ALLOWABLE NUMBER OF ITERATIONS.')
      WRITE (6,62) MAXITN,RELACC,TOLRNC
   62 FORMAT (1H0,'MAXITN =',I4,///,'RELACC =',D28.16,
     1///,'TOLRNC =',D28.16)
      STOP
      END
C
C
      SUBROUTINE RZOUT
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),SCLWT(10,16,3),RHOIN,
     1ZSIN(3)
      DIMENSION ISCLWT(10,16,3)
C
      DO 90 NCOPY=1,5
C
      WRITE(6,10)
   10 FORMAT(1H1,///)
      MARGIN = 4 + NCOPY
      DO 12 IMMGN=5,MARGIN
      WRITE(6,11)
   11 FORMAT(1H )
   12 CONTINUE
```

```
C
      DO 80 ITABLE=1,3
C
      SCALCR = 1000.
      IF (SCLWT(1,16,ITABLE) .GE. 10.) SCALOR = 100.
C
      DO 14 I = 1,10
      DO 14 J = 1,16
      ISCLWT(I,J,ITABLE) = SCALOR*SCLWT(I,J,ITABLE) + 0.5
   14 CONTINUE
C
      IF (SCALOR .EQ. 100.) WRITE(6,16) RHOIN,ZSIN(ITABLE)
   16 FORMAT(1H-,10X,'RHO =',F5.2,6X,'SCALED INITIAL SERVER',
     1' LOAD =',F5.2,12X,'SCALED MEAN WAIT IN HUNDREDTHS')
C
      IF (SCALOR .EQ. 1000.) WRITE(6,20) RHOIN,ZSIN(ITABLE)
   20 FORMAT(1H-,10X,'RHO =',F5.2,6X,'SCALED INITIAL SERVER',
     1' LOAD =',F5.2,11X,'SCALED MEAN WAIT IN THOUSANDTHS')
C
      WRITE (6,25)
   25 FORMAT(1H ,55X,'SCALED EPOCH')
C
      WRITE (6,30) TVAL
   30 FORMAT(1H ,21X,9F5.2,7F5.1)
C
      WRITE (6,35)
   35 FORMAT(1H ,21X,16(1X,'----'))
C
      WRITE (6,40) (ISCLWT(1,J,ITABLE), J=1,16)
   40 FORMAT(1H ,10X,'WIENER      ',16(1X,I4))
C
      WRITE (6,45) (ISCLWT(2,J,ITABLE), J=1,16)
   45 FORMAT(1H ,10X,'MD1 QUEUE  ',16(1X,I4))
C
      DO 60 I=3,9
      WRITE (6,50) KVAL(I-2), (ISCLWT(I,J,ITABLE),J=1,16)
   50 FORMAT(1H ,10X,'ME1 K=',F5.2,16(1X,I4))
   60 CONTINUE
C
      WRITE (6,70) (ISCLWT(10,J,ITABLE), J=1,16)
   70 FORMAT(1H ,10X,'GAMMA INPUT',16(1X,I4))
C
   80 CONTINUE
C
   90 CONTINUE
C
      RETURN
      END
```

COMPUTER PROGRAM FOR APPENDIX TABLES, RHO EQUAL TO ONE

```
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),WAIT(10,16,3),ZIN(3)
      COMMON /LAPLAC/ V(50),M,N
      COMMON /NEWRAP/ TOLENC,MAXITN
C
      M=0
      N=34
      MAXITN=20
      TOLENC=1.D-20
C
      DO 60 IPAGE=1,3
      READ(5,20)  (ZIN(ITABLE), ITABLE=1,3)
      READ(5,20)  (TVAL(J), J=1,6)
      READ(5,20)  (TVAL(J), J=7,12)
      READ(5,20)  (TVAL(J), J=13,16)
   20 FORMAT(6F10.0)
C
      DO 50 ITABLE=1,3
      CALL RZWNR(ITABLE)
      CALL RZMD1(ITABLE)
      CALL RZMEK(ITABLE)
      CALL RZGAM(ITABLE)
   50 CONTINUE
      CALL RZOUT
   60 CONTINUE
      STOP
      END
C
C
      BLOCK DATA
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),WAIT(10,16,3),ZIN(3)
      DATA KVAL/4.,3.,2.,1.5,1.,.5,.2/
      END
C
C
      SUBROUTINE LINV(P,N,T,FA,V,M)
      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      COMMON/LAPLAC/EPOCH(10),APPROX(10,6)
      DIMENSION E(25), F(25), H(25), V(50)
      IF (M.EQ.N) GO TO 50
      NH = N/2
      E(1) = 2
      E(2) = 12
      F(1) = 1
      F(2) = 1
      DO 10 I = 2, NH
      E(I+1) = ( 2.D0*(2*I+1)*E(I) ) / I
      F(I+1) = F(I)*I
   10 CONTINUE
      FNR = NH
```

```
      H(1) = 2/F(NH)
      DO 20 J = 2, NH
      H(J) = ((J**NH)*F(J))/F(NH-J+1)
   20 CONTINUE
      ISIGN = 2*(NH-2*(NH/2))-1
      DO 40 I = 1, N
      V(I) = 0
      JFIRST = (I+1)/2
      JLAST = MIN0(I,NH)
      DO 30 J = JFIRST, JLAST
      V(I) = V(I) + H(J)/(F(I-J+1)*F(2*J-I+1))
   30 CONTINUE
      V(I) = ISIGN*V(I)
      ISIGN = -ISIGN
   40 CONTINUE
      M = N
   50 FA = 0
      A = .693147180559945309417232121458D0 / T
      DO 60 I= 1, N
      FA = FA + V(I)*P(I*A)
   60 CONTINUE
      FA = A*FA
      RETURN
      END
C
C
      SUBROUTINE RZWNR(ITABLE)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),WAIT(10,16,3),ZIN(3)
      COMMON /LAPLAC/ V(50),M,N
      COMMON /ZO/ Z,OSTART
      EXTERNAL PWNR
C
      Z       = ZIN(ITABLE)
      DO 20 J=1,16
      T = TVAL(J)
      CALL LINV(PWNR,N,T,TERM3,V,M)
      WAIT(1,J,ITABLE) = Z + TERM3
   20 CONTINUE
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION PWNR(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /ZO/ Z,OSTART
      OMEGA = DSQRT(2*S)
      PWNR = DEXP(-OMEGA*Z)/(S*OMEGA)
      RETURN
      END
C
C
```

```
      SUBROUTINE EZMD1(ITABLE)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),WAIT(10,16,3),ZIN(3)
      COMMON /LAPLAC/ V(50),M,N
      COMMON /ZO/ Z,OSTART
      EXTERNAL PMD1,FMD1,DFMD1
      OSTART = 1
      Z      = ZIN(ITABLE)
      ELZ = DEXP(-Z)
C
      DO 20 J=1,16
      T = TVAL(J)
      IF (T .LE. Z) EZIT = 0.
      IF (T .GT. Z) CALL LINV(PMD1,N,T,HT,V,M)
      IF (T .GT. Z) EZIT = HT + (T-Z)*ELZ
      WAIT(2,J,ITABLE) = Z + EZIT
   20 CONTINUE
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION PMD1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /ZO/ Z,OSTART
      COMMON SS
      EXTERNAL FOMD1,DFMD1
      SS= S
      CALL ZERO(FOMD1,DFMD1,OSTART,OMEGA)
      OSTART = OMEGA
      PMD1 = DEXP(-OMEGA*Z)/(S*OMEGA) - DEXP(-Z*(1+S))/(S*S)
      RETURN
      END
C
      DOUBLE PRECISION FUNCTION FMD1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /ZO/ Z,OSTART
      FMD1 = S - 1. + DEXP(-S)
      RETURN
      END
C
      DOUBLE PRECISION FUNCTION DFMD1(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /ZO/ Z,OSTART
      DFMD1 = 1.-DEXP(-S)
      RETURN
      END
C
      DOUBLE PRECISION FUNCTION FOMD1(OMEGA)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON SS
      FOMD1 = FMD1(OMEGA) - SS
      RETURN
      END
```

```
      SUBROUTINE RZMEK(ITABLE)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),WAIT(10,16,3),ZIN(3)
      COMMON /LAPLAC/ V(50),M,N
      COMMON /ZOK/ Z,OSTART,K
      EXTERNAL PMEK,FMEK,DFMEK
C
      DO 30 I=3,9
      K       = KVAL(I-2)
      OSTART = 1
      Z       = ZIN(ITABLE)
      ELZ = DEXP(-Z*(K+1)/K)
C
      DO 20 J=1,16
      T = TVAL(J)
      IF (T .LE. Z) EZIT = 0.
      IF (T .GT. Z) CALL LINV(PMEK,N,T,HT,V,M)
      IF (T .GT. Z) EZIT = HT + (T-Z)*ELZ
      WAIT(I,J,ITABLE) = Z + EZIT
   20 CONTINUE
   30 CONTINUE
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION PMEK(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /ZOK/ Z,OSTART,K
      COMMON SS
      EXTERNAL FOMEK,DFMEK
      SS= S
      CALL ZERC(FOMEK,DFMEK,OSTART,OMEGA)
      OSTART = OMEGA
      PMEK = DEXP(-OMEGA*Z)/(S*OMEGA) - DEXP(-Z*(S+(K+1)/K))/(S*S)
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION FMEK(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /ZOK/ Z,OSTART,K
      FMEK = S-(K+1)*(1-((K+1)/(K+1+S))**K)/K
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION DFMEK(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /ZOK/ Z,OSTART,K
      DFMEK = 1-((K+1)/(K+1+S))**(K+1)
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION FOMEK(OMEGA)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON SS
      FOMEK = FMEK(OMEGA) - SS
      RETURN
      END
C
C
      SUBROUTINE RZGAM(ITABLE)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),WAIT(10,16,3),ZIN(3)
      COMMON /LAPLAC/ V(50),M,N
      COMMON /ZO/ Z,OSTART
      EXTERNAL PGAM,FGAM,DFGAM
      OSTART = 1
      Z      = ZIN(ITABLE)
C
      DO 20 J=1,16
      T = TVAL(J)
      IF (T .LE. Z) EZIT = 0.
      IF (T .GT. Z) CALL LINV(PGAM,N,T,EZIT,V,M)
      WAIT(10,J,ITABLE) = Z + EZIT
   20 CONTINUE
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION PGAM(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /ZO/ Z,OSTART
      COMMON SS
      EXTERNAL FOGAM,DFGAM
      SS= S
      CALL ZERO(FOGAM,DFGAM,OSTART,OMEGA)
      OSTART = OMEGA
      PGAM = DEXP(-OMEGA*Z)/(S*OMEGA)
      RETURN
      END
C
      DOUBLE PRECISION FUNCTION FGAM(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /ZO/ Z,OSTART
      FGAM = S-DLOG(1+S)
      RETURN
      END
C
      DOUBLE PRECISION FUNCTION DFGAM(S)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /ZO/ Z,OSTART
      DFGAM = S/(1+S)
      RETURN
      END
```

```
C
      DOUBLE PRECISION FUNCTION FOGAM(OMEGA)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON SS
      FOGAM = FGAM(OMEGA) - SS
      RETURN
      END
C
C
      SUBROUTINE ZERO(F,DF,XSTART,XZERO)
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /NEWRAP/ TOLRNC,MAXITN
      ITERAT = 0
C
      X = XSTART
   10 CONTINUE
      XNEW = X - F(X)/DF(X)
      RELACC = (XNEW-X)/XNEW
      X = XNEW
      ITERAT = ITERAT + 1
      IF (ITERAT .GE. MAXITN) GO TO 50
      IF (DABS(RELACC) .LE. TOLRNC) GO TO 20
      GO TO 10
   20 CONTINUE
      XZERO = X
      RETURN
C
   50 WRITE (6,60)
   60 FORMAT (1H1,'NEWTON-RAPHSON SEARCH IN SUBROUTINE ZERO',
     1' EXCEEDED MAXIMUM ALLOWABLE NUMBER OF ITERATIONS.')
      WRITE (6,62) MAXITN,RELACC,TOLRNC
   62 FORMAT (1H0,'MAXITN =',I4,///,'RELACC =',D28.16,
     1///,'TOLRNC =',D28.16)
      STOP
      END
C
C
      SUBROUTINE EZOUT
      IMPLICIT REAL*8(A-H,K,L,O-Z), INTEGER(I,J,M,N)
      COMMON /INOUT/ TVAL(16),KVAL(7),WAIT(10,16,3),ZIN(3)
      DIMENSION IWAIT(10,16,3)
C
      DO 90 NCOPY=1,5
C
      WRITE(6,10)
   10 FORMAT(1H1,///)
      MARGIN = 6 + NCOPY
      DO 12 IMAGE=7,MARGIN
      WRITE(6,11)
   11 FORMAT(1H )
   12 CONTINUE
C
```

```
      DO 80 ITABLE=1,3
C
      SCALOR = 1000.
      IF (WAIT(1,16,ITABLE) .GE. 10.) SCALOR = 100.
C
      DO 14 I = 1,10
      DO 14 J = 1,16
      IWAIT(I,J,ITABLE) = SCALOR*WAIT(I,J,ITABLE) + 0.5
   14 CONTINUE
C
      WRITE(6,25)
   25 FORMAT(1H-,10X,'PARAMETERS FOR THESE PROCESSES SELECTED',
     1' SO THAT VAR(X(1))=1. AND E(X(1))=0.0, I.E., RHO=1.0')
C
      IF (SCALOR .EQ. 100.) WRITE(6,16) ZIN(ITABLE)
   16 FORMAT(1H ,10X,'MEAN WAIT IN HUNDREDTHS',
     143X,'INITIAL SERVER LOAD = ',F3.1)
C
      IF (SCALOR .EQ. 1000.) WRITE(6,20) ZIN(ITABLE)
   20 FORMAT(1H ,10X,'MEAN WAIT IN THOUSANDTHS',
     142X,'INITIAL SERVER LOAD = ',F3.1)
C
      WRITE (6,30) TVAL
   30 FORMAT(1H ,10X,'EPOCH:',5X,8F5.2,8F5.1)
C
      WRITE (6,35)
   35 FORMAT(1H ,21X,16(1X,'----'))
C
      WRITE (6,40) (IWAIT(1,J,ITABLE), J=1,16)
   40 FORMAT(1H ,10X,'WIENER      ',16(1X,I4))
C
      WRITE (6,45) (IWAIT(2,J,ITABLE), J=1,16)
   45 FORMAT(1H ,10X,'MD1 QUEUE  ',16(1X,I4))
C
      DO 60 I=3,9
      WRITE (6,50) KVAL(I-2), (IWAIT(I,J,ITABLE),J=1,16)
   50 FORMAT(1H ,10X,'ME1 K=',F5.2,16(1X,I4))
   60 CONTINUE
C
      WRITE (6,70) (IWAIT(10,J,ITABLE), J=1,16)
   70 FORMAT(1H ,10X,'GAMMA INPUT',16(1X,I4))
C
   80 CONTINUE
C
   90 CONTINUE
C
      RETURN
      END
```

APPENDIX C

TABLES OF SCALED EXPECTED SERVER LOAD

All tables for a specific value of ρ are grouped together. The tables are arranged in the order shown in this list. Each page contains results for three values of z^*.

ρ		z^*
2.0		
1.5	There are two pages	
1.4	of tables for each	1.0, 0.8, 0.6
1.3	of these five values of ρ.	0.4, 0.2, 0.0
1.2		
1.1		
1.0		
0.9	There are three	4.0, 3.0, 2.0
0.8	pages of tables	1.0, 0.8, 0.6
0.7	for each of these seven values of ρ.	0.4, 0.2, 0.0
0.6		
0.5		

RHO = 2.00 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.60	0.80	1.00	1.20	1.40	1.60	1.80	2.00	2.50	3.0	3.5	4.0	5.0	6.0	7.0	8.0
WIENER	1615	1824	2031	2237	2441	2645	2849	3052	3557	4060	4562	5064	6066	7067	8067	9067
MD1 QUEUE	1600	1800	2000	2203	2404	2605	2806	3007	3509	4010	4511	5012	6012	7013	8013	9013
ME1 K= 4.00	1600	1800	2000	2201	2402	2603	2804	3004	3506	4007	4508	5008	6008	7009	8009	9009
ME1 K= 3.00	1600	1800	2000	2201	2402	2602	2803	3004	3505	4006	4507	5007	6008	7008	8008	9008
ME1 K= 2.00	1600	1800	2000	2201	2401	2602	2803	3003	3504	4005	4506	5006	6006	7006	8007	9007
ME1 K= 1.50	1600	1800	2000	2200	2401	2601	2802	3003	3504	4004	4505	5005	6005	7006	8006	9006
ME1 K= 1.00	1600	1800	2000	2200	2401	2601	2802	3002	3503	4003	4504	5004	6004	7004	8005	9005
ME1 K= 0.50	1600	1800	2000	2200	2400	2601	2801	3001	3502	4002	4503	5003	6003	7003	8003	9003
ME1 K= 0.20	1600	1800	2000	2200	2400	2600	2800	3001	3501	4001	4502	5002	6002	7002	8002	9002
GAMMA INPUT	1600	1800	2000	2200	2400	2600	2800	3000	3501	4001	4501	5001	6001	7001	8001	9001

RHO = 2.00 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.60	0.80	1.00	1.20	1.40	1.60	1.80	2.00	2.50	3.0	3.5	4.0	5.0	6.0	7.0	8.0
WIENER	1433	1645	1855	2063	2269	2474	2678	2882	3388	3892	4395	4897	5899	6900	7900	8901
MD1 QUEUE	1400	1600	1806	2008	2211	2413	2615	2817	3319	3821	4322	4823	5824	6824	7824	8824
ME1 K= 4.00	1400	1600	1803	2005	2207	2409	2610	2811	3314	3815	4316	4816	5817	6817	7817	8818
ME1 K= 3.00	1400	1600	1802	2004	2206	2408	2609	2810	3312	3814	4315	4815	5816	6816	7816	8816
ME1 K= 2.00	1400	1600	1802	2003	2205	2406	2608	2809	3311	3812	4313	4813	5814	6814	7814	8814
ME1 K= 1.50	1400	1600	1801	2003	2204	2405	2607	2807	3309	3810	4311	4811	5812	6812	7812	8812
ME1 K= 1.00	1400	1600	1801	2002	2203	2404	2605	2806	3308	3809	4309	4810	5810	6810	7810	8810
ME1 K= 0.50	1400	1600	1800	2001	2202	2403	2503	2804	3305	3806	4307	4807	5807	6807	7807	8807
ME1 K= 0.20	1400	1600	1800	2001	2201	2402	2602	2803	3304	3804	4305	4805	5805	6805	7805	8805
GAMMA INPUT	1400	1600	1800	2000	2201	2401	2601	2802	3302	3803	4303	4803	5803	6804	7804	8804

RHO = 2.00 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.60	0.80	1.00	1.20	1.40	1.60	1.80	2.00	2.50	3.0	3.5	4.0	5.0	6.0	7.0	8.0
WIENER	1265	1482	1695	1905	2112	2319	2524	2728	3236	3741	4244	4746	5748	6749	7750	8750
MD1 QUEUE	1200	1413	1618	1823	2028	2231	2434	2636	3140	3642	4144	4645	5646	6646	7646	8646
ME1 K= 4.00	1200	1407	1612	1816	2020	2223	2425	2627	3130	3632	4133	4634	5635	6635	7635	8635
ME1 K= 3.00	1200	1406	1611	1815	2018	2221	2423	2625	3128	3630	4131	4632	5632	6633	7633	8633
ME1 K= 2.00	1200	1405	1609	1813	2016	2218	2420	2622	3125	3626	4127	4628	5629	6629	7629	8629
ME1 K= 1.50	1200	1404	1608	1811	2014	2216	2418	2620	3122	3624	4125	4625	5626	6626	7626	8626
ME1 K= 1.00	1200	1403	1606	1809	2011	2214	2415	2617	3119	3620	4121	4622	5622	6623	7623	8623
ME1 K= 0.50	1200	1402	1604	1806	2008	2210	2411	2612	3114	3616	4116	4617	5617	6617	7617	8617
ME1 K= 0.20	1200	1401	1602	1804	2006	2207	2408	2609	3111	3612	4112	4613	5613	6613	7613	8613
GAMMA INPUT	1200	1400	1601	1803	2004	2205	2406	2607	3108	3609	4109	4609	5610	6610	7610	8610

RHO = 2.00 SCALED INITIAL SERVER LOAD = 0.40 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	454	518	583	647	772	892	1008	1122	1343	1559	2085	2599	3614	4619	6623	8624
MD1 QUEUE	450	500	550	600	700	800	917	1028	1240	1450	1966	2475	3483	4486	6487	8488
ME1 K= 4.00	450	500	550	600	700	800	911	1018	1230	1439	1953	2460	3467	4469	6471	8471
ME1 K= 3.00	450	500	550	600	700	800	909	1016	1228	1436	1950	2457	3463	4466	6467	8467
ME1 K= 2.00	450	500	550	600	700	800	908	1014	1224	1432	1945	2452	3458	4460	6461	8461
ME1 K= 1.50	450	500	550	600	700	800	906	1012	1222	1429	1941	2448	3453	4455	6456	8457
ME1 K= 1.00	450	500	550	600	700	800	905	1010	1218	1425	1936	2442	3448	4449	6450	8450
ME1 K= 0.50	450	500	550	600	700	800	903	1007	1214	1420	1929	2434	3439	4440	6441	8441
ME1 K= 0.20	450	500	550	600	700	800	902	1004	1210	1415	1923	2428	3431	4433	6433	8433
GAMMA INPUT	450	500	550	600	700	800	901	1003	1207	1411	1918	2422	3425	4426	6427	8427

RHO = 2.00 SCALED INITIAL SERVER LOAD = 0.20 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	283	369	447	520	655	782	902	1019	1244	1463	1992	2508	3523	4530	6534	8535
MD1 QUEUE	250	300	350	400	537	662	779	889	1107	1321	1841	2351	3360	4364	6365	8366
ME1 K= 4.00	250	300	350	400	529	648	762	874	1091	1303	1821	2330	3338	4341	6342	8342
ME1 K= 3.00	250	300	350	400	527	645	759	870	1087	1298	1816	2325	3333	4335	6337	8337
ME1 K= 2.00	250	300	350	400	524	641	754	865	1081	1292	1808	2317	3324	4327	6328	8328
ME1 K= 1.50	250	300	350	400	522	638	750	861	1076	1287	1803	2311	3318	4320	6322	8322
ME1 K= 1.00	250	300	350	400	518	633	745	855	1069	1280	1795	2302	3309	4311	6312	8312
ME1 K= 0.50	250	300	350	400	514	627	738	846	1059	1269	1782	2289	3295	4297	6298	8298
ME1 K= 0.20	250	300	350	400	510	622	731	839	1051	1259	1771	2277	3282	4284	6284	8285
GAMMA INPUT	250	300	350	400	508	617	725	832	1043	1250	1761	2267	3271	4272	6273	8273

RHO = 2.00 SCALED INITIAL SERVER LOAD = 0.0 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	205	306	392	469	609	738	860	978	1205	1425	1955	2472	3488	4494	6499	8500
MD1 QUEUE	95	182	263	338	475	600	716	829	1051	1266	1787	2298	3308	4311	6313	8314
ME1 K= 4.00	94	179	256	327	459	581	697	811	1030	1243	1763	2273	3282	4285	6286	8286
ME1 K= 3.00	94	178	254	324	455	577	693	806	1025	1238	1757	2267	3275	4278	6280	8280
ME1 K= 2.00	93	176	251	320	449	570	686	799	1017	1230	1748	2257	3265	4268	6269	8270
ME1 K= 1.50	93	174	248	317	445	566	681	793	1011	1223	1741	2250	3257	4260	6261	8262
ME1 K= 1.00	92	172	244	312	439	559	674	785	1002	1214	1731	2239	3246	4249	6250	8250
ME1 K= 0.50	90	168	239	305	430	548	662	773	989	1199	1715	2222	3229	4231	6232	8232
ME1 K= 0.20	89	164	233	299	422	539	651	761	976	1186	1700	2207	3212	4214	6215	8215
GAMMA INPUT	86	160	228	293	414	530	641	751	964	1173	1686	2192	3197	4198	6199	8199

RHO = 1.50 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.30	0.40	0.50	0.60	0.70	0.80	1.00	1.20	1.40	1.6	1.8	2.0	2.5	3.0	5.0	8.0
WIENER	1304	1408	1512	1616	1720	1824	2031	2237	2441	2645	2849	3052	3557	4060	6066	9067
MD1 QUEUE	1300	1400	1500	1601	1703	1804	2007	2210	2413	2615	2817	3019	3522	4024	6027	9028
ME1 K= 4.00	1300	1400	1500	1601	1702	1803	2005	2208	2410	2612	2814	3015	3518	4019	6022	9023
ME1 K= 3.00	1300	1400	1500	1601	1701	1802	2005	2207	2409	2611	2813	3014	3517	4018	6021	9021
ME1 K= 2.00	1300	1400	1500	1600	1701	1802	2004	2206	2408	2610	2811	3013	3515	4017	6019	9020
ME1 K= 1.50	1300	1400	1500	1600	1701	1802	2004	2206	2407	2609	2811	3012	3514	4016	6018	9018
ME1 K= 1.00	1300	1400	1500	1600	1701	1801	2003	2205	2406	2608	2809	3010	3513	4014	6016	9017
ME1 K= 0.50	1300	1400	1500	1600	1700	1801	2002	2204	2405	2606	2807	3008	3510	4011	6013	9014
ME1 K= 0.20	1300	1400	1500	1600	1700	1800	2001	2203	2404	2605	2806	3007	3508	4009	6011	9011
GAMMA INPUT	1300	1400	1500	1600	1700	1800	2001	2202	2403	2604	2805	3005	3507	4008	6009	9009

RHO = 1.50 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.30	0.40	0.50	0.60	0.70	0.80	1.00	1.20	1.40	1.6	1.8	2.0	2.5	3.0	5.0	8.0
WIENER	1112	1219	1327	1433	1540	1645	1855	2063	2269	2474	2678	2882	3388	3892	5899	8901
MD1 QUEUE	1100	1200	1303	1406	1510	1613	1818	2023	2227	2430	2633	2835	3339	3842	5846	8847
ME1 K= 4.00	1100	1200	1302	1404	1507	1609	1814	2018	2222	2425	2627	2829	3333	3835	5839	8839
ME1 K= 3.00	1100	1200	1302	1404	1506	1609	1813	2017	2221	2424	2626	2828	3331	3834	5837	8838
ME1 K= 2.00	1100	1200	1301	1403	1505	1608	1812	2016	2219	2422	2624	2826	3329	3831	5834	8835
ME1 K= 1.50	1100	1200	1301	1403	1505	1607	1811	2015	2218	2420	2622	2824	3327	3829	5832	8833
ME1 K= 1.00	1100	1200	1301	1402	1504	1606	1809	2013	2216	2418	2620	2822	3325	3827	5830	8830
ME1 K= 0.50	1100	1200	1300	1401	1503	1604	1807	2010	2213	2415	2617	2818	3321	3823	5825	8826
ME1 K= 0.20	1100	1200	1300	1401	1502	1603	1806	2008	2211	2412	2614	2815	3318	3819	5822	8822
GAMMA INPUT	1100	1200	1300	1400	1501	1602	1804	2007	2209	2410	2612	2813	3315	3816	5818	8819

RHO = 1.50 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.30	0.40	0.50	0.60	0.70	0.80	1.00	1.20	1.40	1.6	1.8	2.0	2.5	3.0	5.0	8.0
WIENER	930	1044	1155	1265	1374	1482	1695	1905	2112	2319	2524	2728	3236	3741	5748	8750
MD1 QUEUE	900	1009	1115	1221	1327	1432	1641	1848	2054	2258	2462	2665	3170	3673	5678	8679
ME1 K= 4.00	900	1005	1111	1216	1322	1426	1634	1841	2046	2250	2453	2656	3160	3663	5668	8669
ME1 K= 3.00	900	1005	1110	1215	1320	1425	1633	1839	2044	2248	2451	2654	3158	3661	5665	8666
ME1 K= 2.00	900	1004	1109	1214	1318	1423	1630	1836	2041	2245	2448	2650	3155	3657	5661	8662
ME1 K= 1.50	900	1003	1108	1212	1317	1421	1628	1834	2039	2242	2445	2648	3152	3655	5658	8659
ME1 K= 1.00	900	1002	1106	1211	1315	1419	1626	1831	2036	2239	2442	2644	3148	3651	5654	8655
ME1 K= 0.50	900	1002	1105	1208	1312	1416	1622	1827	2031	2234	2437	2639	3142	3645	5648	8648
ME1 K= 0.20	900	1001	1103	1206	1310	1413	1618	1823	2026	2229	2432	2634	3137	3639	5642	8642
GAMMA INPUT	900	1000	1102	1205	1308	1410	1615	1819	2023	2225	2427	2629	3132	3634	5637	8637

RHO = 1.50 SCALED INITIAL SERVER LOAD = 0.40 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	454	518	583	647	772	892	1008	1122	1343	1559	2085	2599	3614	4619	6623	8624
MD1 QUEUE	450	500	550	600	722	835	947	1058	1274	1487	2006	2517	3527	4531	6533	8533
ME1 K= 4.00	450	500	550	600	715	828	939	1049	1264	1476	1994	2504	3513	4517	6519	8519
ME1 K= 3.00	450	500	550	600	714	826	938	1047	1262	1473	1991	2501	3510	4514	6516	8516
ME1 K= 2.00	450	500	550	600	712	824	935	1044	1258	1469	1987	2496	3505	4508	6510	8511
ME1 K= 1.50	450	500	550	600	711	822	932	1041	1256	1466	1983	2492	3501	4504	6506	8506
ME1 K= 1.00	450	500	550	600	709	820	929	1038	1252	1462	1978	2487	3495	4498	6500	8500
ME1 K= 0.50	450	500	550	600	707	816	925	1033	1245	1455	1970	2479	3486	4489	6490	8491
ME1 K= 0.20	450	500	550	600	705	813	921	1028	1240	1449	1963	2471	3478	4480	6482	8482
GAMMA INPUT	450	500	550	600	703	810	917	1024	1235	1443	1956	2464	3470	4472	6474	8474

RHO = 1.50 SCALED INITIAL SERVER LOAD = 0.20 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	283	369	447	520	655	782	902	1019	1244	1463	1992	2508	3523	4530	6534	8535
MD1 QUEUE	250	300	383	454	581	704	821	935	1156	1371	1894	2407	3418	4422	6425	8425
ME1 K= 4.00	250	300	375	443	571	692	809	922	1142	1357	1879	2390	3401	4405	6407	8408
ME1 K= 3.00	250	300	374	441	569	690	806	919	1139	1353	1875	2386	3397	4401	6403	8403
ME1 K= 2.00	250	300	371	438	565	685	801	914	1134	1348	1869	2380	3390	4394	6396	8396
ME1 K= 1.50	250	300	369	436	562	682	798	911	1130	1343	1864	2375	3385	4388	6390	8391
ME1 K= 1.00	250	300	367	432	558	677	793	905	1124	1337	1857	2368	3377	4380	6382	8383
ME1 K= 0.50	250	300	363	428	551	670	785	897	1114	1327	1846	2356	3365	4368	6370	8370
ME1 K= 0.20	250	300	361	423	546	663	777	889	1106	1318	1836	2345	3353	4356	6358	8358
GAMMA INPUT	250	300	358	420	541	657	770	881	1097	1309	1826	2335	3342	4345	6346	8347

RHO = 1.50 SCALED INITIAL SERVER LOAD = 0.0 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	205	306	392	469	609	738	860	978	1205	1425	1955	2472	3488	4494	6499	8500
MD1 QUEUE	131	232	315	387	524	649	769	884	1108	1324	1848	2361	3373	4378	6381	8381
ME1 K= 4.00	127	222	303	376	511	636	755	870	1092	1307	1831	2343	3354	4358	6361	8361
ME1 K= 3.00	125	220	301	374	508	632	751	866	1088	1303	1827	2339	3349	4354	6356	8356
ME1 K= 2.00	124	217	297	369	503	627	746	860	1082	1297	1820	2331	3342	4346	6348	8348
ME1 K= 1.50	122	214	294	366	499	623	741	856	1077	1292	1814	2326	3336	4340	6342	8342
ME1 K= 1.00	120	211	289	361	494	617	735	849	1070	1284	1806	2317	3327	4331	6333	8333
ME1 K= 0.50	117	206	283	354	485	608	725	838	1058	1272	1793	2304	3313	4316	6318	8319
ME1 K= 0.20	114	201	277	347	477	599	715	828	1048	1261	1781	2291	3300	4303	6304	8305
GAMMA INPUT	111	196	271	341	470	591	706	819	1037	1250	1769	2279	3287	4290	6291	8291

RHO = 1.40 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.5	2.0	2.5	3.0	4.0	6.0	8.0
WIENER	1201	1304	1408	1512	1616	1720	1824	1928	2031	2544	3052	3557	4060	5064	7067	9067
MD1 QUEUE	1200	1300	1400	1501	1603	1704	1806	1908	2010	2518	3023	3526	4028	5031	7032	9033
ME1 K= 4.00	1200	1300	1400	1501	1602	1703	1805	1906	2008	2514	3019	3522	4024	5026	7027	9028
ME1 K= 3.00	1200	1300	1400	1500	1601	1703	1804	1906	2007	2514	3018	3521	4023	5025	7026	9027
ME1 K= 2.00	1200	1300	1400	1500	1601	1702	1804	1905	2006	2513	3017	3520	4021	5023	7025	9025
ME1 K= 1.50	1200	1300	1400	1500	1601	1702	1803	1905	2006	2512	3016	3518	4020	5022	7023	9024
ME1 K= 1.00	1200	1300	1400	1500	1601	1702	1803	1904	2005	2510	3014	3517	4018	5020	7021	9022
ME1 K= 0.50	1200	1300	1400	1500	1600	1701	1802	1903	2004	2509	3012	3514	4016	5018	7019	9019
ME1 K= 0.20	1200	1300	1400	1500	1600	1701	1801	1902	2003	2507	3010	3512	4014	5015	7016	9016
GAMMA INPUT	1200	1300	1400	1500	1600	1700	1801	1902	2002	2506	3009	3510	4012	5013	7014	9014

RHO = 1.40 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.5	2.0	2.5	3.0	4.0	6.0	8.0
WIENER	1005	1112	1219	1327	1433	1540	1645	1750	1855	2372	2882	3388	3892	4897	6900	8901
MD1 QUEUE	1000	1100	1202	1306	1409	1513	1617	1720	1823	2334	2841	3346	3848	4851	6853	8854
ME1 K= 4.00	1000	1100	1201	1304	1407	1510	1613	1716	1819	2329	2836	3340	3842	4845	6847	8847
ME1 K= 3.00	1000	1100	1201	1304	1406	1509	1612	1715	1818	2328	2834	3338	3841	4843	6845	8845
ME1 K= 2.00	1000	1100	1201	1303	1406	1508	1611	1714	1816	2326	2832	3336	3838	4841	6842	8843
ME1 K= 1.50	1000	1100	1201	1303	1405	1508	1610	1713	1815	2325	2831	3334	3836	4839	6840	8841
ME1 K= 1.00	1000	1100	1200	1302	1404	1507	1609	1712	1814	2323	2828	3332	3834	4836	6838	8838
ME1 K= 0.50	1000	1100	1200	1301	1403	1505	1607	1709	1811	2320	2825	3328	3830	4832	6833	8834
ME1 K= 0.20	1000	1100	1200	1301	1402	1504	1606	1708	1810	2317	2822	3324	3826	4828	6829	8830
GAMMA INPUT	1000	1100	1200	1301	1402	1503	1605	1706	1808	2314	2819	3321	3823	4825	6826	8826

RHO = 1.40 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.5	2.0	2.5	3.0	4.0	6.0	8.0
WIENER	815	930	1044	1155	1265	1374	1482	1589	1695	2216	2728	3236	3741	4746	6749	8750
MD1 QUEUE	800	905	1013	1120	1227	1333	1439	1544	1648	2164	2673	3179	3682	4686	6688	8689
ME1 K= 4.00	800	903	1009	1116	1222	1328	1433	1538	1642	2157	2665	3170	3673	4677	6679	8679
ME1 K= 3.00	800	903	1009	1115	1221	1327	1432	1536	1640	2155	2663	3168	3671	4675	6677	8677
ME1 K= 2.00	800	902	1007	1113	1219	1325	1430	1534	1638	2152	2660	3165	3668	4671	6673	8673
ME1 K= 1.50	800	902	1007	1112	1218	1323	1428	1532	1636	2150	2658	3162	3665	4668	6670	8671
ME1 K= 1.00	800	901	1006	1111	1216	1321	1426	1530	1633	2147	2654	3159	3661	4664	6666	8666
ME1 K= 0.50	800	901	1004	1109	1213	1318	1422	1526	1629	2142	2649	3153	3656	4658	6660	8660
ME1 K= 0.20	800	900	1003	1107	1211	1315	1419	1522	1626	2137	2644	3148	3650	4652	6654	8654
GAMMA INPUT	800	900	1002	1105	1209	1313	1416	1519	1622	2133	2639	3143	3645	4647	6648	8649

RHO = 1.40 SCALED INITIAL SERVER LOAD = 0.40 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	454	518	583	647	772	892	1008	1122	1343	1559	2085	2599	3614	4619	6623	8624
MD1 QUEUE	450	500	550	611	727	843	956	1067	1284	1497	2017	2529	3539	4543	6546	8547
ME1 K= 4.00	450	500	550	607	722	836	948	1059	1275	1487	2007	2517	3527	4531	6533	8534
ME1 K= 3.00	450	500	550	606	721	835	947	1057	1273	1485	2004	2514	3524	4528	6530	8531
ME1 K= 2.00	450	500	550	605	719	832	944	1054	1270	1481	2000	2510	3520	4523	6525	8526
ME1 K= 1.50	450	500	550	604	718	830	942	1052	1267	1478	1997	2507	3516	4519	6522	8522
ME1 K= 1.00	450	500	550	603	716	828	939	1048	1263	1474	1992	2502	3511	4514	6516	8516
ME1 K= 0.50	450	500	550	602	713	824	934	1043	1257	1468	1985	2494	3502	4505	6507	8507
ME1 K= 0.20	450	500	550	601	710	820	930	1038	1252	1462	1978	2486	3494	4497	6499	8499
GAMMA INPUT	450	500	550	601	708	817	926	1034	1246	1456	1971	2479	3487	4489	6491	8491

RHO = 1.40 SCALED INITIAL SERVER LOAD = 0.20 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	283	369	447	520	655	782	902	1019	1244	1463	1992	2508	3523	4530	6534	8535
MD1 QUEUE	250	317	394	461	592	715	833	948	1169	1385	1909	2422	3434	4438	6441	8442
ME1 K= 4.00	250	313	385	454	583	705	822	936	1157	1372	1895	2407	3419	4423	6426	8426
ME1 K= 3.00	250	312	384	452	581	703	820	933	1154	1369	1892	2404	3415	4419	6422	8422
ME1 K= 2.00	250	310	381	449	577	699	815	929	1150	1364	1886	2398	3409	4413	6415	8416
ME1 K= 1.50	250	309	379	447	574	696	812	926	1146	1360	1882	2394	3404	4408	6411	8411
ME1 K= 1.00	250	307	377	443	571	691	807	921	1140	1354	1876	2387	3397	4401	6403	8404
ME1 K= 0.50	250	305	373	438	564	684	800	913	1132	1345	1866	2377	3386	4390	6392	8392
ME1 K= 0.20	250	304	369	434	559	678	793	905	1123	1336	1856	2366	3376	4379	6381	8381
GAMMA INPUT	250	303	366	430	553	672	786	898	1116	1328	1847	2357	3365	4369	6370	8371

RHO = 1.40 SCALED INITIAL SERVER LOAD = 0.0 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	205	306	392	469	609	738	860	978	1205	1425	1955	2472	3488	4494	6499	8500
MD1 QUEUE	143	244	325	401	537	663	783	899	1123	1339	1865	2378	3391	4396	6399	8399
ME1 K= 4.00	138	234	316	390	525	651	770	886	1109	1325	1849	2362	3374	4378	6381	8382
ME1 K= 3.00	136	232	314	388	523	648	767	883	1105	1321	1846	2358	3370	4374	6377	8377
ME1 K= 2.00	134	229	310	384	518	643	762	877	1100	1316	1839	2352	3363	4367	6370	8371
ME1 K= 1.50	133	227	307	380	515	639	758	873	1095	1311	1835	2347	3358	4362	6365	8365
ME1 K= 1.00	130	223	303	376	510	634	753	867	1089	1304	1827	2339	3350	4354	6357	8357
ME1 K= 0.50	127	218	297	369	502	625	743	858	1079	1294	1816	2327	3338	4341	6344	8344
ME1 K= 0.20	123	213	291	362	494	617	735	849	1069	1283	1805	2316	3325	4329	6331	8331
GAMMA INPUT	120	209	286	356	487	609	726	840	1060	1274	1794	2305	3314	4317	6319	8319

RHO = 1.30 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.15	0.20	0.25	0.30	0.40	0.50	0.60	0.70	0.80	0.9	1.0	1.5	2.0	4.0	6.0	8.0
WIENER	1150	1201	1252	1304	1408	1512	1616	1720	1824	1928	2031	2544	3052	5064	7067	9067
MD1 QUEUE	1150	1200	1250	1300	1401	1503	1605	1707	1809	1911	2013	2522	3028	5036	7038	9039
ME1 K= 4.00	1150	1200	1250	1300	1401	1502	1603	1705	1807	1909	2011	2519	3024	5032	7034	9034
ME1 K= 3.00	1150	1200	1250	1300	1400	1502	1603	1705	1807	1909	2011	2518	3023	5031	7033	9033
ME1 K= 2.00	1150	1200	1250	1300	1400	1501	1603	1704	1806	1908	2010	2517	3022	5030	7031	9032
ME1 K= 1.50	1150	1200	1250	1300	1400	1501	1602	1704	1806	1908	2009	2516	3021	5029	7030	9030
ME1 K= 1.00	1150	1200	1250	1300	1400	1501	1602	1704	1805	1907	2008	2515	3020	5027	7028	9028
ME1 K= 0.50	1150	1200	1250	1300	1400	1501	1502	1703	1804	1906	2007	2513	3017	5024	7025	9026
ME1 K= 0.20	1150	1200	1250	1300	1400	1500	1601	1702	1803	1905	2006	2512	3015	5022	7023	9023
GAMMA INPUT	1150	1200	1250	1300	1400	1500	1601	1702	1803	1904	2005	2510	3014	5019	7020	9021

RHO = 1.30 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.15	0.20	0.25	0.30	0.40	0.50	0.60	0.70	0.80	0.9	1.0	1.5	2.0	4.0	6.0	8.0
WIENER	952	1005	1058	1112	1219	1327	1433	1540	1645	1750	1855	2372	2882	4897	6900	8901
MD1 QUEUE	950	1000	1050	1101	1205	1309	1413	1518	1621	1725	1828	2341	2849	4860	6862	8862
ME1 K= 4.00	950	1000	1050	1101	1203	1307	1411	1515	1618	1722	1825	2336	2843	4854	6856	8856
ME1 K= 3.00	950	1000	1050	1101	1203	1307	1410	1514	1617	1721	1824	2335	2842	4852	6854	8855
ME1 K= 2.00	950	1000	1050	1100	1203	1306	1409	1513	1616	1719	1822	2333	2840	4850	6852	8852
ME1 K= 1.50	950	1000	1050	1100	1202	1305	1409	1512	1615	1718	1821	2332	2839	4848	6850	8851
ME1 K= 1.00	950	1000	1050	1100	1202	1305	1408	1511	1614	1717	1820	2330	2837	4846	6848	8848
ME1 K= 0.50	950	1000	1050	1100	1201	1304	1406	1509	1612	1715	1817	2327	2833	4842	6843	8844
ME1 K= 0.20	950	1000	1050	1100	1201	1303	1405	1508	1610	1713	1815	2324	2830	4838	6840	8840
GAMMA INPUT	950	1000	1050	1100	1201	1302	1404	1507	1609	1711	1813	2322	2827	4835	6836	8836

RHO = 1.30 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.15	0.20	0.25	0.30	0.40	0.50	0.60	0.70	0.80	0.9	1.0	1.5	2.0	4.0	6.0	8.0
WIENER	759	816	873	930	1044	1155	1265	1374	1482	1589	1695	2216	2728	4746	6749	8750
MD1 QUEUE	750	802	855	909	1018	1126	1234	1341	1447	1552	1657	2174	2684	4697	6700	8700
ME1 K= 4.00	750	801	853	907	1015	1122	1229	1336	1442	1547	1651	2167	2677	4689	6692	8692
ME1 K= 3.00	750	801	853	906	1014	1121	1228	1335	1440	1545	1650	2166	2675	4687	6690	8690
ME1 K= 2.00	750	800	853	906	1013	1120	1227	1333	1438	1543	1648	2163	2672	4684	6686	8687
ME1 K= 1.50	750	800	852	905	1012	1119	1225	1331	1437	1542	1646	2161	2670	4682	6684	8684
ME1 K= 1.00	750	800	852	904	1011	1117	1224	1329	1435	1539	1643	2158	2667	4678	6680	8681
ME1 K= 0.50	750	800	851	903	1009	1115	1221	1326	1431	1535	1639	2153	2662	4673	6674	8675
ME1 K= 0.20	750	800	851	902	1007	1113	1218	1323	1428	1532	1636	2149	2657	4667	6669	8669
GAMMA INPUT	750	800	850	902	1006	1111	1216	1321	1425	1529	1632	2145	2652	4662	6664	8664

RHO = 1.30 SCALED INITIAL SERVER LOAD = 0.40 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	454	518	583	647	772	892	1008	1122	1343	1559	2085	2599	3614	4619	6623	8624
MD1 QUEUE	450	500	558	617	736	852	966	1077	1296	1509	2030	2542	3554	4558	6561	8562
ME1 K= 4.00	450	500	555	613	731	846	959	1070	1288	1501	2021	2533	3543	4548	6550	8551
ME1 K= 3.00	450	500	554	612	730	845	958	1069	1286	1499	2019	2530	3541	4545	6548	8548
ME1 K= 2.00	450	500	553	611	728	843	955	1066	1283	1496	2016	2527	3537	4541	6543	8544
ME1 K= 1.50	450	500	553	610	726	841	953	1064	1281	1493	2013	2524	3534	4538	6540	8541
ME1 K= 1.00	450	500	552	609	724	838	950	1061	1277	1489	2009	2519	3529	4533	6535	8536
ME1 K= 0.50	450	500	551	607	721	834	946	1056	1272	1483	2002	2512	3522	4525	6527	8528
ME1 K= 0.20	450	500	551	605	718	831	942	1051	1266	1478	1996	2505	3514	4518	6520	8520
GAMMA INPUT	450	500	551	604	716	828	938	1047	1262	1472	1990	2499	3508	4511	6513	8513

RHO = 1.30 SCALED INITIAL SERVER LOAD = 0.20 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	283	369	447	520	655	782	902	1019	1244	1463	1992	2508	3523	4530	6534	8535
MD1 QUEUE	250	331	403	473	605	729	847	962	1184	1401	1926	2439	3452	4457	6460	8460
ME1 K= 4.00	250	324	397	466	597	720	838	952	1174	1390	1914	2427	3439	4444	6447	8447
ME1 K= 3.00	250	322	395	465	595	718	836	950	1172	1387	1911	2424	3436	4441	6443	8444
ME1 K= 2.00	250	321	393	462	592	714	832	946	1168	1383	1907	2419	3431	4435	6438	8439
ME1 K= 1.50	250	319	391	460	589	712	829	943	1164	1380	1903	2415	3427	4431	6434	8435
ME1 K= 1.00	250	317	389	457	586	708	825	939	1160	1375	1898	2410	3421	4425	6428	8428
ME1 K= 0.50	250	314	385	452	580	701	818	932	1152	1367	1889	2401	3412	4416	6418	8419
ME1 K= 0.20	250	312	381	448	575	696	812	925	1145	1359	1881	2392	3402	4406	6409	8409
GAMMA INPUT	250	310	377	444	570	690	806	919	1138	1352	1873	2383	3393	4397	6399	8400

RHO = 1.30 SCALED INITIAL SERVER LOAD = 0.0 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	205	306	392	469	609	738	860	978	1205	1425	1955	2472	3488	4494	6499	8500
MD1 QUEUE	158	256	339	414	551	678	799	915	1139	1357	1883	2397	3410	4416	6419	8419
ME1 K= 4.00	151	248	331	405	542	668	788	904	1128	1345	1870	2384	3397	4401	6404	8405
ME1 K= 3.00	149	247	329	403	539	666	785	901	1125	1342	1867	2381	3393	4398	6401	8402
ME1 K= 2.00	147	244	326	400	536	661	781	897	1120	1337	1862	2375	3388	4392	6395	8396
ME1 K= 1.50	145	242	323	397	533	658	778	894	1117	1333	1858	2371	3383	4388	6391	8391
ME1 K= 1.00	143	238	319	393	528	654	773	888	1111	1328	1852	2365	3377	4381	6384	8384
ME1 K= 0.50	139	233	314	387	521	646	765	880	1103	1318	1842	2355	3366	4370	6373	8373
ME1 K= 0.20	136	229	308	381	515	639	758	872	1094	1310	1833	2345	3356	4360	6362	8363
GAMMA INPUT	133	224	303	375	508	632	750	865	1086	1301	1824	2335	3346	4350	6352	8352

RHO = 1.20 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.50	0.60	0.8	1.0	2.0	3.0	4.0	6.0	8.0
WIENER	1100	1150	1201	1252	1304	1356	1408	1512	1616	1824	2031	3052	4060	5064	7067	9067
MD1 QUEUE	1100	1150	1200	1250	1301	1351	1402	1505	1607	1813	2018	3034	4041	5044	7046	9046
ME1 K= 4.00	1100	1150	1200	1250	1300	1351	1402	1504	1606	1811	2016	3031	4037	5040	7042	9043
ME1 K= 3.00	1100	1150	1200	1250	1300	1351	1402	1504	1606	1811	2015	3030	4037	5039	7041	9042
ME1 K= 2.00	1100	1150	1200	1250	1300	1351	1401	1503	1605	1810	2015	3029	4035	5038	7040	9040
ME1 K= 1.50	1100	1150	1200	1250	1300	1351	1401	1503	1605	1810	2014	3028	4034	5037	7039	9039
ME1 K= 1.00	1100	1150	1200	1250	1300	1351	1401	1503	1605	1809	2013	3027	4033	5036	7037	9038
ME1 K= 0.50	1100	1150	1200	1250	1300	1350	1401	1502	1604	1808	2012	3025	4031	5033	7035	9035
ME1 K= 0.20	1100	1150	1200	1250	1300	1350	1401	1502	1603	1807	2011	3023	4029	5031	7032	9033
GAMMA INPUT	1100	1150	1200	1250	1300	1350	1400	1501	1603	1806	2010	3021	4027	5029	7030	9031

RHO = 1.20 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.50	0.60	0.8	1.0	2.0	3.0	4.0	6.0	8.0
WIENER	900	952	1005	1058	1112	1165	1219	1327	1433	1645	1855	2882	3892	4897	6900	8901
MD1 QUEUE	900	950	1000	1052	1104	1156	1208	1313	1418	1628	1835	2857	3866	4870	6872	8872
ME1 K= 4.00	900	950	1000	1051	1103	1155	1207	1311	1416	1625	1832	2853	3861	4865	6867	8868
ME1 K= 3.00	900	950	1000	1051	1102	1154	1206	1311	1416	1624	1831	2852	3860	4864	6866	8867
ME1 K= 2.00	900	950	1000	1051	1102	1154	1206	1310	1415	1623	1830	2851	3859	4862	6864	8865
ME1 K= 1.50	900	950	1000	1051	1102	1154	1206	1310	1414	1622	1829	2849	3857	4861	6863	8863
ME1 K= 1.00	900	950	1000	1051	1102	1153	1205	1309	1413	1621	1828	2848	3855	4858	6860	8861
ME1 K= 0.50	900	950	1000	1050	1101	1153	1204	1308	1412	1619	1826	2845	3852	4855	6857	8857
ME1 K= 0.20	900	950	1000	1050	1101	1152	1203	1307	1410	1618	1824	2842	3849	4852	6854	8854
GAMMA INPUT	900	950	1000	1050	1101	1152	1203	1306	1409	1616	1822	2839	3846	4849	6850	8851

RHO = 1.20 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.50	0.60	0.8	1.0	2.0	3.0	4.0	6.0	8.0
WIENER	703	759	816	873	930	987	1044	1155	1265	1482	1695	2728	3741	4746	6749	8750
MD1 QUEUE	700	751	805	859	914	970	1025	1134	1242	1456	1667	2696	3706	4711	6713	8714
ME1 K= 4.00	700	751	804	858	912	967	1022	1131	1239	1452	1663	2690	3700	4704	6707	8708
ME1 K= 3.00	700	751	803	857	912	966	1021	1130	1238	1451	1661	2689	3699	4703	6705	8706
ME1 K= 2.00	700	751	803	857	911	965	1020	1129	1236	1449	1660	2687	3696	4700	6703	8703
ME1 K= 1.50	700	750	803	856	910	965	1019	1128	1235	1448	1658	2685	3695	4698	6701	8701
ME1 K= 1.00	700	750	802	856	909	964	1018	1126	1234	1446	1656	2682	3692	4696	6698	8699
ME1 K= 0.50	700	750	802	855	908	962	1016	1124	1231	1443	1653	2678	3687	4691	6693	8694
ME1 K= 0.20	700	750	801	854	907	961	1014	1122	1229	1440	1650	2674	3683	4686	6689	8689
GAMMA INPUT	700	750	801	853	906	959	1013	1120	1226	1438	1646	2670	3679	4682	6684	8685

RHO = 1.20 SCALED INITIAL SERVER LOAD = 0.40 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.30	0.40	0.6	0.8	1.0	2.0	4.0	6.0	8.0
WIENER	420	442	466	492	518	583	647	772	892	1122	1343	1559	2599	4619	6623	8624
MD1 QUEUE	420	440	460	480	504	564	625	746	863	1090	1309	1523	2558	4575	6578	8579
ME1 K= 4.00	420	440	460	480	502	561	622	742	858	1084	1303	1517	2551	4567	6570	8571
ME1 K= 3.00	420	440	460	480	502	561	621	741	857	1083	1302	1515	2549	4565	6568	8569
ME1 K= 2.00	420	440	460	480	502	560	620	739	855	1081	1299	1513	2546	4562	6565	8566
ME1 K= 1.50	420	440	460	480	501	559	619	738	854	1079	1297	1511	2544	4560	6563	8563
ME1 K= 1.00	420	440	460	480	501	558	617	736	852	1077	1294	1508	2540	4556	6559	8559
ME1 K= 0.50	420	440	460	480	501	557	615	733	848	1073	1290	1503	2535	4550	6553	8553
ME1 K= 0.20	420	440	460	480	500	556	614	730	845	1069	1286	1498	2529	4544	6547	8547
GAMMA INPUT	420	440	460	480	500	555	612	728	842	1065	1282	1494	2524	4538	6541	8541

RHO = 1.20 SCALED INITIAL SERVER LOAD = 0.20 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.30	0.40	0.6	0.8	1.0	2.0	4.0	6.0	8.0
WIENER	228	265	301	336	369	447	520	655	782	1019	1244	1463	2508	4530	6534	8535
MD1 QUEUE	220	240	277	308	340	416	487	619	744	978	1202	1418	2459	4477	6481	8482
ME1 K= 4.00	220	240	273	305	336	411	481	613	737	971	1194	1410	2450	4468	6471	8472
ME1 K= 3.00	220	240	272	304	335	410	480	612	735	969	1192	1408	2447	4465	6469	8469
ME1 K= 2.00	220	240	271	302	333	408	478	609	733	966	1189	1405	2444	4462	6465	8465
ME1 K= 1.50	220	240	270	301	332	406	476	607	731	964	1186	1403	2441	4459	6462	8462
ME1 K= 1.00	220	240	269	300	330	404	473	604	728	960	1183	1399	2437	4454	6457	8458
ME1 K= 0.50	220	240	267	297	327	400	469	600	723	955	1177	1393	2430	4446	6449	8450
ME1 K= 0.20	220	240	266	295	325	397	466	595	718	950	1171	1386	2423	4439	6442	8443
GAMMA INPUT	220	240	264	293	322	394	462	591	713	944	1166	1381	2416	4432	6435	8435

RHO = 1.20 SCALED INITIAL SERVER LOAD = 0.0 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.30	0.40	0.6	0.8	1.0	2.0	4.0	6.0	8.0
WIENER	123	181	227	269	306	392	469	609	738	978	1205	1425	2472	4494	6499	8500
MD1 QUEUE	91	147	193	233	271	354	430	568	695	933	1158	1376	2419	4438	6442	8442
ME1 K= 4.00	87	142	188	228	265	348	423	561	688	925	1150	1367	2409	4428	6431	8432
ME1 K= 3.00	86	141	186	226	263	346	422	559	686	923	1148	1365	2406	4425	6429	8429
ME1 K= 2.00	85	139	184	224	261	344	419	556	683	920	1144	1362	2402	4421	6424	8425
ME1 K= 1.50	83	138	183	223	259	342	417	554	680	917	1141	1359	2399	4418	6421	8422
ME1 K= 1.00	82	136	180	220	256	339	414	550	677	913	1137	1355	2395	4413	6416	8416
ME1 K= 0.50	80	133	177	216	252	334	409	545	671	907	1131	1348	2387	4404	6407	8408
ME1 K= 0.20	78	130	173	212	248	330	404	540	665	901	1124	1341	2379	4396	6399	8400
GAMMA INPUT	76	127	170	209	244	325	399	534	660	895	1118	1334	2372	4389	6391	8392

RHO = 1.10 SCALED INITIAL SERVER LOAD = 4.00 SCALED MEAN WAIT IN HUNDREDTHS

	SCALED EPOCH															
	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.5	2.0	3.0	4.0	6.0	10.0	20.0
WIENER	420	430	440	450	460	470	480	490	500	550	600	700	800	1000	1400	2400
MD1 QUEUE	420	430	440	450	460	470	480	490	500	550	600	700	800	1000	1400	2400
ME1 K= 4.00	420	430	440	450	460	470	480	490	500	550	600	700	800	1000	1400	2400
ME1 K= 3.00	420	430	440	450	460	470	480	490	500	550	600	700	800	1000	1400	2400
ME1 K= 2.00	420	430	440	450	460	470	480	490	500	550	600	700	800	1000	1400	2400
ME1 K= 1.50	420	430	440	450	460	470	480	490	500	550	600	700	800	1000	1400	2400
ME1 K= 1.00	420	430	440	450	460	470	480	490	500	550	600	700	800	1000	1400	2400
ME1 K= 0.50	420	430	440	450	460	470	480	490	500	550	600	700	800	1000	1400	2400
ME1 K= 0.20	420	430	440	450	460	470	480	490	500	550	600	700	800	1000	1400	2400
GAMMA INPUT	420	430	440	450	460	470	480	490	500	550	600	700	800	1000	1400	2400

RHO = 1.10 SCALED INITIAL SERVER LOAD = 3.00 SCALED MEAN WAIT IN HUNDREDTHS

	SCALED EPOCH															
	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.5	2.0	3.0	4.0	6.0	10.0	20.0
WIENER	320	330	340	350	360	370	380	390	400	450	500	600	700	900	1300	2300
MD1 QUEUE	320	330	340	350	360	370	380	390	400	450	500	600	700	900	1300	2300
ME1 K= 4.00	320	330	340	350	360	370	380	390	400	450	500	600	700	900	1300	2300
ME1 K= 3.00	320	330	340	350	360	370	380	390	400	450	500	600	700	900	1300	2300
ME1 K= 2.00	320	330	340	350	360	370	380	390	400	450	500	600	700	900	1300	2300
ME1 K= 1.50	320	330	340	350	360	370	380	390	400	450	500	600	700	900	1300	2300
ME1 K= 1.00	320	330	340	350	360	370	380	390	400	450	500	600	700	900	1300	2300
ME1 K= 0.50	320	330	340	350	360	370	380	390	400	450	500	600	700	900	1300	2300
ME1 K= 0.20	320	330	340	350	360	370	380	390	400	450	500	600	700	900	1300	2300
GAMMA INPUT	320	330	340	350	360	370	380	390	400	450	500	600	700	900	1300	2300

RHO = 1.10 SCALED INITIAL SERVER LOAD = 2.00 SCALED MEAN WAIT IN HUNDREDTHS

	SCALED EPOCH															
	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.5	2.0	3.0	4.0	6.0	10.0	20.0
WIENER	220	230	240	250	260	270	280	290	300	350	400	501	601	801	1201	2201
MD1 QUEUE	220	230	240	250	260	270	280	290	300	350	400	500	601	801	1201	2201
ME1 K= 4.00	220	230	240	250	260	270	280	290	300	350	400	500	601	801	1201	2201
ME1 K= 3.00	220	230	240	250	260	270	280	290	300	350	400	500	601	801	1201	2201
ME1 K= 2.00	220	230	240	250	260	270	280	290	300	350	400	500	600	801	1201	2201
ME1 K= 1.50	220	230	240	250	260	270	280	290	300	350	400	500	600	801	1201	2201
ME1 K= 1.00	220	230	240	250	260	270	280	290	300	350	400	500	600	801	1201	2201
ME1 K= 0.50	220	230	240	250	260	270	280	290	300	350	400	500	600	801	1201	2201
ME1 K= 0.20	220	230	240	250	260	270	280	290	300	350	400	500	600	800	1200	2200
GAMMA INPUT	220	230	240	250	260	270	280	290	300	350	400	500	600	800	1200	2200

RHO = 1.10 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	1050	1100	1150	1201	1304	1408	1512	1616	1824	2031	2544	3052	4060	5064	7067	9067
MD1 QUEUE	1050	1100	1150	1200	1302	1404	1508	1611	1818	2023	2535	3042	4049	5053	7055	9055
ME1 K= 4.00	1050	1100	1150	1200	1302	1404	1507	1610	1817	2022	2533	3040	4047	5051	7053	9053
ME1 K= 3.00	1050	1100	1150	1200	1301	1404	1507	1610	1816	2022	2533	3039	4047	5050	7052	9053
ME1 K= 2.00	1050	1100	1150	1200	1301	1404	1507	1610	1816	2021	2532	3039	4046	5049	7051	9052
ME1 K= 1.50	1050	1100	1150	1200	1301	1403	1506	1609	1815	2021	2531	3038	4045	5049	7051	9051
ME1 K= 1.00	1050	1100	1150	1200	1301	1403	1506	1609	1815	2020	2531	3037	4044	5048	7050	9050
ME1 K= 0.50	1050	1100	1150	1200	1301	1403	1506	1608	1814	2019	2529	3036	4043	5046	7048	9049
ME1 K= 0.20	1050	1100	1150	1200	1301	1403	1505	1608	1813	2018	2528	3035	4041	5044	7046	9047
GAMMA INPUT	1050	1100	1150	1200	1301	1402	1505	1607	1813	2018	2527	3033	4040	5043	7045	9045

RHO = 1.10 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	850	900	952	1005	1112	1219	1327	1433	1645	1855	2372	2882	3892	4897	6900	8901
MD1 QUEUE	850	900	951	1002	1107	1213	1319	1425	1635	1844	2359	2868	3878	4882	6884	8885
ME1 K= 4.00	850	900	950	1002	1106	1212	1318	1423	1634	1842	2357	2866	3875	4879	6882	8882
ME1 K= 3.00	850	900	950	1002	1106	1212	1317	1423	1633	1841	2356	2865	3874	4878	6881	8881
ME1 K= 2.00	850	900	950	1001	1106	1211	1317	1422	1632	1841	2355	2864	3873	4877	6880	8880
ME1 K= 1.50	850	900	950	1001	1105	1211	1317	1422	1632	1840	2355	2863	3872	4876	6879	8879
ME1 K= 1.00	850	900	950	1001	1105	1210	1316	1421	1631	1839	2353	2862	3871	4875	6877	8878
ME1 K= 0.50	850	900	950	1001	1105	1210	1315	1420	1630	1838	2352	2860	3869	4873	6875	8876
ME1 K= 0.20	850	900	950	1001	1104	1209	1314	1419	1628	1836	2350	2858	3867	4871	6873	8874
GAMMA INPUT	850	900	950	1001	1104	1208	1313	1418	1627	1835	2348	2857	3865	4868	6871	8871

RHO = 1.10 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.5	2.0	3.0	4.0	6.0	8.0
WIENER	650	703	759	816	930	1044	1155	1265	1482	1695	2216	2728	3741	4746	6749	8750
MD1 QUEUE	650	701	754	810	921	1033	1143	1253	1468	1679	2199	2710	3722	4726	6730	8730
ME1 K= 4.00	650	701	754	809	920	1031	1141	1250	1465	1677	2196	2707	3718	4723	6726	8727
ME1 K= 3.00	650	701	754	808	920	1031	1141	1250	1465	1676	2195	2706	3717	4722	6725	8726
ME1 K= 2.00	650	700	753	808	919	1030	1140	1249	1464	1675	2194	2705	3716	4721	6723	8724
ME1 K= 1.50	650	700	753	808	918	1029	1139	1248	1463	1674	2193	2704	3715	4719	6722	8723
ME1 K= 1.00	650	700	753	807	918	1028	1138	1247	1461	1673	2191	2702	3713	4718	6721	8721
ME1 K= 0.50	650	700	752	806	917	1027	1137	1245	1459	1671	2189	2700	3710	4715	6718	8718
ME1 K= 0.20	550	700	752	806	916	1026	1135	1244	1458	1668	2187	2697	3708	4712	6715	8715
GAMMA INPUT	650	700	752	805	915	1024	1134	1242	1456	1666	2184	2695	3705	4709	6712	8713

RHO = 1.10 SCALED INITIAL SERVER LOAD = 0.40 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.40	0.60	0.8	1.0	2.0	3.0	4.0	6.0	8.0
WIENER	420	442	466	492	518	583	647	892	1122	1343	1559	2599	3614	4619	6623	8624
MD1 QUEUE	420	440	462	485	510	572	635	876	1104	1325	1540	2577	3590	4595	6599	8600
ME1 K= 4.00	420	440	461	484	508	571	633	873	1101	1321	1536	2573	3586	4591	6594	8595
ME1 K= 3.00	420	440	461	484	508	570	632	873	1100	1320	1535	2572	3585	4590	6593	8594
ME1 K= 2.00	420	440	461	484	508	569	631	871	1099	1319	1534	2570	3583	4588	6591	8592
ME1 K= 1.50	420	440	461	483	507	569	631	871	1098	1318	1532	2569	3581	4586	6590	8591
ME1 K= 1.00	420	440	461	483	507	568	630	869	1097	1316	1531	2567	3579	4584	6588	8588
ME1 K= 0.50	420	440	460	483	506	567	628	867	1094	1313	1528	2563	3576	4581	6584	8585
ME1 K= 0.20	420	440	460	482	505	566	627	865	1092	1311	1525	2560	3572	4577	6580	8581
GAMMA INPUT	420	440	460	482	505	564	625	863	1089	1308	1522	2557	3569	4574	6577	8577

RHO = 1.10 SCALED INITIAL SERVER LOAD = 0.20 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.40	0.60	0.8	1.0	2.0	3.0	4.0	6.0	8.0
WIENER	228	265	301	336	369	447	520	782	1019	1244	1463	2508	3523	4530	6534	8535
MD1 QUEUE	220	253	287	321	354	430	502	761	997	1221	1439	2481	3496	4501	6505	8506
ME1 K= 4.00	220	251	285	319	351	427	499	757	993	1217	1434	2476	3490	4496	6500	8501
ME1 K= 3.00	220	251	285	318	350	427	498	756	992	1216	1433	2475	3489	4495	6498	8499
ME1 K= 2.00	220	250	284	317	349	425	497	755	990	1214	1431	2473	3487	4492	6496	8497
ME1 K= 1.50	220	250	283	316	348	424	496	754	989	1213	1430	2471	3485	4491	6494	8495
ME1 K= 1.00	220	249	282	315	347	423	494	752	987	1210	1428	2469	3483	4488	6492	8492
ME1 K= 0.50	220	248	280	313	345	421	492	749	983	1207	1424	2465	3478	4484	6487	8488
ME1 K= 0.20	220	247	279	311	343	418	489	746	980	1204	1420	2461	3474	4479	6483	8484
GAMMA INPUT	220	246	278	310	341	416	487	743	977	1200	1417	2457	3470	4475	6479	8479

RHO = 1.10 SCALED INITIAL SERVER LOAD = 0.0 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.40	0.60	0.8	1.0	2.0	3.0	4.0	6.0	8.0
WIENER	123	181	227	269	306	392	469	738	978	1205	1425	2472	3488	4494	6499	8500
MD1 QUEUE	106	163	209	250	287	372	448	715	954	1180	1399	2443	3458	4464	6468	8469
ME1 K= 4.00	103	160	206	246	284	368	444	711	949	1175	1394	2438	3452	4458	6462	8463
ME1 K= 3.00	103	159	205	246	283	367	443	710	948	1174	1393	2436	3451	4457	6461	8462
ME1 K= 2.00	101	158	204	244	281	365	441	708	946	1172	1390	2434	3449	4455	6458	8459
ME1 K= 1.50	101	157	203	243	280	364	440	706	945	1171	1389	2432	3447	4453	6456	8457
ME1 K= 1.00	99	155	201	242	279	363	438	704	943	1168	1386	2430	3444	4450	6453	8454
ME1 K= 0.50	97	153	199	239	276	360	435	701	939	1164	1382	2425	3439	4445	6449	8449
ME1 K= 0.20	95	151	196	236	273	357	432	698	935	1161	1378	2421	3435	4440	6444	8445
GAMMA INPUT	94	149	194	234	271	354	429	694	932	1157	1375	2417	3430	4436	6439	8440

PARAMETERS FOR THESE PROCESSES SELECTED SO THAT VAR(X(1))=1. AND E(X(1))=0.0, I.E., RHO=1.0
MEAN WAIT IN THOUSANDTHS INITIAL SERVER LOAD = 4.0

EPOCH:	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.0	15.0	20.0	25.0	30.0	40.0	60.0	80.0
WIENER	4002	4012	4034	4066	4105	4151	4201	4254	4310	4606	4907	5202	5486	6023	6987	7838
MD1 QUEUE	4000	4000	4000	4014	4037	4070	4109	4153	4202	4475	4766	5054	5334	5866	6825	7675
ME1 K= 4.00	4000	4000	4000	4009	4029	4058	4095	4137	4184	4452	4739	5026	5305	5836	6794	7644
ME1 K= 3.00	4000	4000	4000	4008	4027	4055	4091	4133	4179	4446	4733	5019	5298	5829	6786	7636
ME1 K= 2.00	4000	4000	4000	4006	4024	4051	4086	4127	4172	4437	4722	5008	5286	5816	6774	7623
ME1 K= 1.50	4000	4000	4000	4005	4022	4048	4082	4122	4167	4429	4714	4999	5277	5806	6763	7612
ME1 K= 1.00	4000	4000	4000	4004	4019	4043	4076	4115	4159	4418	4701	4985	5263	5792	6748	7597
ME1 K= 0.50	4000	4000	4000	4003	4014	4036	4067	4104	4146	4400	4681	4963	5240	5768	6723	7571
ME1 K= 0.20	4000	4000	4000	4002	4011	4030	4058	4094	4134	4383	4661	4942	5218	5744	6698	7545
GAMMA INPUT	4000	4000	4000	4001	4008	4025	4051	4084	4123	4366	4641	4921	5195	5720	6673	7520

PARAMETERS FOR THESE PROCESSES SELECTED SO THAT VAR(X(1))=1. AND E(X(1))=0.0, I.E., RHO=1.0
MEAN WAIT IN THOUSANDTHS INITIAL SERVER LOAD = 3.0

EPOCH:	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.0	15.0	20.0	25.0	30.0	40.0	60.0	80.0
WIENER	3017	3059	3117	3186	3261	3339	3419	3500	3580	3974	4342	4687	5010	5604	6638	7534
MD1 QUEUE	3000	3000	3036	3087	3149	3218	3291	3366	3443	3824	4187	4528	4849	5441	6474	7369
ME1 K= 4.00	3000	3000	3026	3073	3131	3198	3269	3343	3418	3796	4158	4498	4819	5410	6442	7337
ME1 K= 3.00	3000	3000	3024	3069	3127	3193	3264	3337	3412	3789	4150	4491	4811	5402	6434	7329
ME1 K= 2.00	3000	3000	3021	3064	3120	3185	3255	3328	3402	3778	4139	4479	4799	5389	6421	7316
ME1 K= 1.50	3000	3000	3019	3060	3115	3179	3248	3321	3395	3769	4129	4469	4789	5379	6411	7305
ME1 K= 1.00	3000	3000	3016	3054	3108	3170	3238	3310	3383	3756	4115	4454	4774	5364	6395	7289
ME1 K= 0.50	3000	3000	3011	3046	3096	3156	3223	3293	3365	3734	4092	4430	4750	5339	6369	7263
ME1 K= 0.20	3000	3000	3008	3038	3085	3143	3208	3276	3348	3714	4070	4407	4726	5314	6344	7237
GAMMA INPUT	3000	3000	3006	3032	3076	3131	3194	3261	3331	3693	4048	4384	4702	5289	6318	7211

PARAMETERS FOR THESE PROCESSES SELECTED SO THAT VAR(X(1))=1. AND E(X(1))=0.0, I.E., RHO=1.0
MEAN WAIT IN THOUSANDTHS INITIAL SERVER LOAD = 2.0

EPOCH:	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.0	15.0	20.0	25.0	30.0	40.0	60.0	80.0
WIENER	2101	2213	2333	2454	2572	2687	2799	2907	3012	3493	3919	4304	4658	5296	6385	7314
MD1 QUEUE	2000	2092	2200	2311	2424	2536	2644	2751	2854	3332	3756	4140	4494	5131	6219	7148
ME1 K= 4.00	2000	2077	2178	2287	2398	2509	2617	2722	2825	3302	3725	4109	4462	5099	6187	7116
ME1 K= 3.00	2000	2073	2173	2281	2392	2502	2610	2716	2818	3294	3718	4101	4454	5091	6179	7108
ME1 K= 2.00	2000	2067	2164	2272	2382	2491	2599	2704	2807	3282	3705	4088	4441	5078	6166	7094
ME1 K= 1.50	2000	2062	2158	2265	2374	2483	2590	2695	2798	3272	3695	4078	4431	5068	6155	7084
ME1 K= 1.00	2000	2056	2149	2254	2362	2471	2577	2682	2784	3258	3680	4063	4416	5052	6140	7068
ME1 K= 0.50	2000	2046	2135	2237	2344	2451	2557	2660	2762	3234	3656	4038	4391	5027	6113	7041
ME1 K= 0.20	2000	2038	2121	2221	2326	2431	2536	2639	2740	3211	3632	4014	4366	5001	6088	7015
GAMMA INPUT	2000	2031	2109	2206	2308	2413	2517	2619	2720	3188	3608	3990	4341	4976	6062	6989

PARAMETERS FOR THESE PROCESSES SELECTED SO THAT VAR(X(1))=1. AND E(X(1))=0.0, I.E., RHO=1.0
MEAN WAIT IN THOUSANDTHS INITIAL SERVER LOAD = 1.0

EPOCH:	0.60	0.80	1.00	1.50	2.00	3.00	4.00	5.00	6.0	8.0	10.0	15.0	20.0	40.0	60.0	80.0
WIENER	1072	1118	1167	1286	1399	1606	1791	1960	2115	2396	2648	3193	3657	5109	6232	7181
MD1 QUEUE	1000	1000	1000	1146	1238	1444	1629	1796	1951	2231	2483	3027	3491	4943	6065	7014
ME1 K= 4.00	1000	1000	1000	1115	1219	1419	1600	1767	1921	2201	2452	2995	3459	4911	6033	6982
ME1 K= 3.00	1000	1000	1000	1109	1213	1412	1593	1760	1914	2194	2445	2988	3452	4903	6025	6974
ME1 K= 2.00	1000	1000	1000	1101	1204	1401	1582	1748	1902	2181	2432	2975	3439	4890	6012	6961
ME1 K= 1.50	1000	1000	1000	1095	1197	1393	1573	1739	1893	2172	2422	2965	3428	4879	6001	6950
ME1 K= 1.00	1000	1000	1000	1087	1187	1381	1561	1726	1879	2158	2408	2950	3413	4863	5985	6934
ME1 K= 0.50	1000	1000	1000	1075	1171	1362	1540	1704	1857	2134	2384	2925	3388	4838	5959	6907
ME1 K= 0.20	1000	1000	1000	1064	1156	1344	1520	1683	1835	2112	2361	2901	3364	4812	5933	6881
GAMMA INPUT	1000	1000	1000	1055	1143	1327	1501	1663	1814	2090	2338	2878	3339	4787	5907	6855

PARAMETERS FOR THESE PROCESSES SELECTED SO THAT VAR(X(1))=1. AND E(X(1))=0.0, I.E., RHO=1.0
MEAN WAIT IN THOUSANDTHS INITIAL SERVER LOAD = 0.8

EPOCH:	0.60	0.80	1.00	1.50	2.00	3.00	4.00	5.00	6.0	8.0	10.0	15.0	20.0	40.0	60.0	80.0
WIENER	921	981	1040	1179	1304	1527	1722	1897	2058	2346	2603	3156	3625	5087	6213	7165
MD1 QUEUE	800	800	882	1026	1140	1362	1557	1732	1892	2180	2437	2989	3459	4920	6047	6998
ME1 K= 4.00	800	800	865	996	1117	1335	1528	1702	1862	2150	2407	2958	3427	4888	6014	6966
ME1 K= 3.00	800	800	861	990	1111	1328	1521	1695	1855	2143	2399	2951	3419	4880	6006	6958
ME1 K= 2.00	800	800	855	981	1101	1317	1510	1684	1843	2130	2387	2938	3407	4867	5993	6944
ME1 K= 1.50	800	800	850	974	1093	1309	1501	1674	1834	2121	2377	2928	3396	4856	5982	6934
ME1 K= 1.00	800	800	844	964	1082	1296	1488	1661	1820	2106	2362	2913	3381	4841	5967	6918
ME1 K= 0.50	800	800	835	949	1064	1276	1466	1639	1797	2083	2338	2888	3356	4815	5940	6891
ME1 K= 0.20	800	800	828	935	1048	1257	1446	1617	1775	2060	2315	2864	3331	4789	5914	6865
GAMMA INPUT	800	800	821	923	1032	1240	1427	1597	1754	2038	2292	2840	3307	4764	5889	6839

PARAMETERS FOR THESE PROCESSES SELECTED SO THAT VAR(X(1))=1. AND E(X(1))=0.0, I.E., RHO=1.0
MEAN WAIT IN THOUSANDTHS INITIAL SERVER LOAD = 0.6

EPOCH:	0.60	0.80	1.00	1.50	2.00	3.00	4.00	5.00	6.0	8.0	10.0	15.0	20.0	40.0	60.0	80.0
WIENER	795	868	937	1092	1228	1464	1667	1848	2013	2307	2568	3127	3600	5069	6199	7153
MD1 QUEUE	600	700	781	925	1066	1299	1501	1682	1846	2141	2402	2961	3434	4902	6032	6986
ME1 K= 4.00	600	684	753	902	1037	1270	1472	1652	1816	2110	2371	2929	3402	4870	6000	6953
ME1 K= 3.00	600	680	747	896	1030	1263	1465	1645	1809	2103	2364	2922	3394	4862	5992	6945
ME1 K= 2.00	600	673	738	887	1020	1252	1453	1633	1797	2091	2351	2909	3381	4849	5979	6932
ME1 K= 1.50	600	668	732	879	1012	1243	1444	1624	1788	2081	2341	2899	3371	4839	5968	6921
ME1 K= 1.00	600	662	723	868	1000	1230	1431	1610	1774	2067	2327	2884	3356	4823	5952	6905
ME1 K= 0.50	600	652	710	852	981	1210	1409	1588	1751	2043	2303	2859	3331	4797	5926	6879
ME1 K= 0.20	600	644	598	836	964	1191	1389	1566	1729	2020	2279	2835	3306	4771	5900	6853
GAMMA INPUT	600	636	688	822	948	1172	1369	1546	1708	1998	2257	2811	3282	4746	5874	6827

PARAMETERS FOR THESE PROCESSES SELECTED SO THAT VAR(X(1))=1. AND E(X(1))=0.0, I.E., RHO=1.0
MEAN WAIT IN THOUSANDTHS INITIAL SERVER LOAD = 0.4

EPOCH:	0.20	0.40	0.60	0.80	1.00	1.50	2.00	3.00	4.0	6.0	8.0	10.0	20.0	40.0	60.0	80.0
WIENER	491	602	699	784	861	1029	1173	1419	1628	1980	2279	2543	3583	5056	6189	7144
MD1 QUEUE	400	400	521	621	703	857	1010	1253	1461	1814	2113	2377	3416	4890	6022	6977
ME1 K= 4.00	400	400	508	595	670	836	979	1224	1432	1784	2082	2346	3384	4858	5990	6945
ME1 K= 3.00	400	400	504	589	665	830	973	1217	1425	1776	2075	2338	3377	4850	5982	6936
ME1 K= 2.00	400	400	498	580	655	820	962	1205	1413	1764	2062	2326	3364	4836	5968	6923
ME1 K= 1.50	400	400	493	574	648	812	954	1197	1404	1755	2053	2316	3353	4826	5958	6912
ME1 K= 1.00	400	400	486	565	638	801	942	1184	1390	1741	2038	2301	3338	4810	5942	6896
ME1 K= 0.50	400	400	476	552	623	783	923	1163	1369	1718	2015	2277	3313	4784	5916	6870
ME1 K= 0.20	400	400	468	540	610	767	905	1143	1348	1696	1992	2254	3288	4759	5890	6844
GAMMA INPUT	400	400	460	530	597	752	888	1124	1328	1674	1970	2231	3264	4734	5864	6818

PARAMETERS FOR THESE PROCESSES SELECTED SO THAT VAR(X(1))=1. AND E(X(1))=0.0, I.E., RHO=1.0
MEAN WAIT IN THOUSANDTHS INITIAL SERVER LOAD = 0.2

EPOCH:	0.20	0.40	0.60	0.80	1.00	1.50	2.00	3.00	4.0	6.0	8.0	10.0	20.0	40.0	60.0	80.0
WIENER	392	530	639	731	814	990	1140	1391	1604	1961	2262	2528	3572	5049	6182	7138
MD1 QUEUE	200	348	470	569	651	822	974	1225	1437	1794	2096	2362	3405	4882	6016	6972
ME1 K= 4.00	200	338	448	540	622	797	945	1196	1408	1764	2065	2331	3374	4850	5983	6939
ME1 K= 3.00	200	335	443	535	616	790	939	1189	1401	1757	2058	2323	3366	4842	5975	6931
ME1 K= 2.00	200	330	435	526	606	780	928	1178	1389	1745	2045	2311	3353	4829	5962	6918
ME1 K= 1.50	200	326	430	519	599	772	920	1169	1380	1735	2036	2301	3343	4818	5952	6907
ME1 K= 1.00	200	321	422	510	589	761	907	1156	1366	1721	2021	2286	3327	4803	5936	6891
ME1 K= 0.50	200	312	410	496	574	743	888	1135	1345	1698	1998	2262	3302	4777	5909	6865
ME1 K= 0.20	200	304	399	483	559	726	870	1115	1324	1676	1975	2239	3278	4751	5883	6838
GAMMA INPUT	200	297	389	471	546	711	853	1096	1304	1655	1953	2216	3253	4726	5858	6812

PARAMETERS FOR THESE PROCESSES SELECTED SO THAT VAR(X(1))=1. AND E(X(1))=0.0, I.E., RHO=1.0
MEAN WAIT IN THOUSANDTHS INITIAL SERVER LOAD = 0.0

EPOCH:	0.20	0.40	0.60	0.80	1.00	1.50	2.00	3.00	4.0	6.0	8.0	10.0	20.0	40.0	60.0	80.0
WIENER	357	505	618	714	798	977	1128	1382	1596	1954	2257	2523	3568	5046	6180	7136
MD1 QUEUE	181	330	451	551	632	810	962	1215	1429	1788	2090	2356	3402	4880	6014	6970
ME1 K= 4.00	177	316	428	522	606	784	934	1187	1400	1758	2060	2326	3370	4848	5981	6937
ME1 K= 3.00	175	312	423	517	600	777	927	1180	1393	1750	2052	2318	3362	4840	5973	6929
ME1 K= 2.00	173	307	415	508	591	767	917	1168	1381	1738	2040	2306	3349	4826	5960	6916
ME1 K= 1.50	171	302	409	502	583	759	908	1159	1372	1729	2030	2296	3339	4816	5949	6905
ME1 K= 1.00	168	296	401	492	573	748	896	1146	1358	1715	2016	2281	3324	4800	5934	6889
ME1 K= 0.50	164	287	389	478	558	730	877	1126	1337	1692	1992	2257	3299	4774	5907	6863
ME1 K= 0.20	159	278	378	465	543	713	859	1106	1316	1670	1969	2234	3274	4749	5881	6837
GAMMA INPUT	155	270	367	453	530	698	842	1087	1296	1648	1947	2211	3250	4723	5856	6811

RHO = 0.90 SCALED INITIAL SERVER LOAD = 4.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.20	0.40	0.60	0.80	1.00	1.50	2.00	2.50	3.00	4.0	5.0	6.0	7.0	8.0	10.0	12.0
WIENER	3800	3600	3400	3200	3001	2514	2073	1701	1404	1001	774	649	582	545	514	504
MD1 QUEUE	3800	3600	3400	3200	3000	2511	2056	1693	1396	996	772	649	582	546	514	505
ME1 K= 4.00	3800	3600	3400	3200	3000	2511	2065	1691	1394	995	771	649	582	546	514	505
ME1 K= 3.00	3800	3600	3400	3200	3000	2510	2065	1691	1394	994	771	649	582	546	514	505
ME1 K= 2.00	3800	3600	3400	3200	3000	2510	2064	1690	1393	994	771	649	582	546	514	505
ME1 K= 1.50	3800	3600	3400	3200	3000	2510	2064	1689	1393	994	771	649	582	546	514	505
ME1 K= 1.00	3800	3600	3400	3200	3000	2510	2063	1689	1392	993	771	649	582	546	515	505
ME1 K= 0.50	3800	3600	3400	3200	3000	2509	2062	1687	1390	992	770	649	583	546	515	505
ME1 K= 0.20	3800	3600	3400	3200	3000	2509	2061	1686	1389	991	770	649	583	546	515	505
GAMMA INPUT	3800	3600	3400	3200	3000	2509	2060	1684	1388	990	770	649	583	546	515	505

RHO = 0.90 SCALED INITIAL SERVER LOAD = 3.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.20	0.40	0.60	0.80	1.00	1.50	2.00	2.50	3.00	4.0	5.0	6.0	7.0	8.0	10.0	12.0
WIENER	2800	2600	2400	2203	2013	1596	1278	1049	887	694	599	551	527	514	504	501
MD1 QUEUE	2800	2600	2400	2202	2010	1588	1269	1041	882	692	599	552	527	515	504	501
ME1 K= 4.00	2800	2600	2400	2202	2009	1587	1267	1040	880	692	599	552	528	515	504	501
ME1 K= 3.00	2800	2600	2400	2202	2009	1586	1267	1039	880	692	599	552	528	515	505	501
ME1 K= 2.00	2800	2600	2400	2202	2009	1586	1266	1039	880	691	599	552	528	515	505	501
ME1 K= 1.50	2800	2600	2400	2201	2009	1585	1266	1038	879	691	599	552	528	515	505	501
ME1 K= 1.00	2800	2600	2400	2201	2009	1584	1265	1037	879	691	599	552	528	515	505	501
ME1 K= 0.50	2800	2600	2400	2201	2008	1583	1263	1036	878	691	598	552	528	515	505	501
ME1 K= 0.20	2800	2600	2400	2201	2008	1582	1262	1035	877	690	598	552	528	515	505	501
GAMMA INPUT	2800	2600	2400	2201	2007	1580	1260	1033	876	690	598	552	528	515	505	501

RHO = 0.90 SCALED INITIAL SERVER LOAD = 2.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.20	0.40	0.60	0.80	1.00	1.50	2.00	2.50	3.00	4.0	5.0	6.0	7.0	8.0	10.0	12.0
WIENER	1800	1602	1418	1258	1126	891	750	664	609	550	524	512	506	503	501	500
MD1 QUEUE	1800	1601	1413	1250	1116	882	744	659	606	550	524	512	506	503	501	500
ME1 K= 4.00	1800	1601	1412	1249	1114	880	742	658	606	549	524	512	506	503	501	500
ME1 K= 3.00	1800	1601	1412	1248	1114	880	742	658	606	549	524	512	506	503	501	500
ME1 K= 2.00	1800	1601	1411	1248	1113	879	741	658	605	549	524	512	506	503	501	500
ME1 K= 1.50	1800	1601	1411	1247	1112	878	741	658	605	549	524	512	506	503	501	500
ME1 K= 1.00	1800	1600	1411	1246	1111	877	740	657	605	549	524	512	506	503	501	500
ME1 K= 0.50	1800	1600	1410	1245	1110	876	739	656	605	549	524	512	506	503	501	500
ME1 K= 0.20	1800	1600	1410	1244	1108	874	738	656	604	549	524	512	506	503	501	500
GAMMA INPUT	1800	1600	1409	1243	1107	873	737	655	604	549	524	512	506	503	501	500

RHO = 0.90 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.50	0.6	0.8	1.0	1.5	2.0	3.0	5.0
WIENER	950	900	853	812	775	744	717	694	657	629	590	565	532	517	506	501
MD1 QUEUE	950	900	851	806	767	734	707	683	645	618	580	556	526	514	504	501
ME1 K= 4.00	950	900	851	805	766	733	705	681	643	615	578	555	525	513	504	501
ME1 K= 3.00	950	900	851	805	766	732	704	680	643	615	577	554	525	513	504	500
ME1 K= 2.00	950	900	851	805	765	732	703	679	642	614	577	554	525	512	504	500
ME1 K= 1.50	950	900	850	805	765	731	703	679	641	613	576	553	524	512	504	500
ME1 K= 1.00	950	900	850	804	764	730	702	678	640	612	575	552	524	512	503	500
ME1 K= 0.50	950	900	850	804	763	729	700	676	638	610	573	551	523	511	503	500
ME1 K= 0.20	950	900	850	803	762	728	699	674	636	608	572	549	522	511	503	500
GAMMA INPUT	950	900	850	803	761	726	697	672	634	606	570	548	521	510	503	500

RHO = 0.90 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.50	0.6	0.8	1.0	1.5	2.0	3.0	5.0
WIENER	750	703	663	632	608	589	574	561	543	531	516	508	500	498	498	499
MD1 QUEUE	750	700	657	623	597	577	561	549	532	520	507	500	495	495	496	499
ME1 K= 4.00	750	700	656	621	595	575	559	547	529	518	505	498	494	494	496	499
ME1 K= 3.00	750	700	656	621	594	574	559	546	529	517	504	498	494	494	496	499
ME1 K= 2.00	750	700	655	620	594	573	558	545	528	516	504	497	493	493	496	499
ME1 K= 1.50	750	700	655	620	593	573	557	545	527	516	503	497	493	493	496	499
ME1 K= 1.00	750	700	655	619	592	571	556	543	526	515	502	496	492	493	496	499
ME1 K= 0.50	750	700	654	618	590	570	554	541	524	513	501	495	491	492	495	499
ME1 K= 0.20	750	700	654	617	589	568	552	540	522	511	499	494	491	492	495	499
GAMMA INPUT	750	700	653	616	587	566	550	538	520	509	497	492	490	491	495	498

RHO = 0.90 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.50	0.6	0.8	1.0	1.5	2.0	3.0	5.0
WIENER	551	514	491	477	468	463	460	458	457	458	462	467	478	485	493	498
MD1 QUEUE	550	507	480	465	456	450	448	446	446	448	454	460	473	482	491	498
ME1 K= 4.00	550	505	478	462	453	448	445	444	444	446	453	459	472	481	491	498
ME1 K= 3.00	550	505	478	462	453	447	445	443	444	446	452	459	472	481	491	498
ME1 K= 2.00	550	505	477	461	452	446	444	443	443	445	452	458	472	481	491	498
ME1 K= 1.50	550	505	476	460	451	446	443	442	442	445	451	458	471	480	491	498
ME1 K= 1.00	550	504	475	459	450	444	442	441	441	444	450	457	471	480	490	498
ME1 K= 0.50	550	503	474	457	448	442	440	439	439	442	449	456	470	480	490	497
ME1 K= 0.20	550	503	472	455	446	440	438	437	438	440	447	455	469	479	490	497
GAMMA INPUT	550	502	471	454	444	438	436	435	436	439	446	454	469	479	490	497

RHO = 0.90 SCALED INITIAL SERVER LOAD = 0.40 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.01	0.02	0.03	0.04	0.06	0.08	0.10	0.15	0.20	0.3	0.4	0.6	1.0	2.0	3.0	4.0
WIENER	390	380	372	365	357	353	351	354	360	374	388	411	442	477	490	495
MD1 QUEUE	390	380	370	360	347	341	338	340	346	362	377	402	435	474	488	494
ME1 K= 4.00	390	380	370	360	346	339	336	337	344	359	375	400	434	473	488	494
ME1 K= 3.00	390	380	370	360	346	339	336	337	343	359	374	400	434	473	488	494
ME1 K= 2.00	390	380	370	360	345	338	335	336	342	358	373	399	433	473	488	494
ME1 K= 1.50	390	380	370	360	345	337	334	335	341	357	373	398	433	473	487	494
ME1 K= 1.00	390	380	370	360	344	336	333	334	340	356	371	397	432	472	487	494
ME1 K= 0.50	390	380	370	360	344	335	331	332	338	354	370	396	431	472	487	494
ME1 K= 0.20	390	380	370	360	343	334	329	330	336	352	368	394	430	471	487	494
GAMMA INPUT	390	380	370	360	342	332	328	328	334	350	366	393	429	471	486	493

RHO = 0.90 SCALED INITIAL SERVER LOAD = 0.20 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.01	0.02	0.03	0.04	0.06	0.08	0.10	0.15	0.20	0.3	0.4	0.6	1.0	2.0	3.0	4.0
WIENER	192	193	198	204	218	231	243	269	290	324	349	385	428	473	488	494
MD1 QUEUE	190	180	184	190	203	216	228	255	277	312	339	377	423	470	487	494
ME1 K= 4.00	190	180	182	187	200	213	225	252	275	310	337	376	421	469	486	493
ME1 K= 3.00	190	180	181	186	199	212	225	252	274	310	337	375	421	469	486	493
ME1 K= 2.00	190	180	181	185	198	211	224	251	273	309	336	374	421	469	486	493
ME1 K= 1.50	190	180	180	185	197	210	223	250	273	308	335	374	420	469	486	493
ME1 K= 1.00	190	180	179	184	196	209	221	249	271	307	334	373	420	468	486	493
ME1 K= 0.50	190	180	178	182	194	207	219	247	269	305	332	372	419	468	485	493
ME1 K= 0.20	190	180	177	180	192	205	217	245	267	303	331	370	418	467	485	493
GAMMA INPUT	190	180	176	179	190	203	215	243	266	302	329	369	416	467	485	493

RHO = 0.90 SCALED INITIAL SERVER LOAD = 0.0 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.01	0.02	0.03	0.04	0.06	0.08	0.10	0.15	0.20	0.3	0.4	0.6	1.0	2.0	3.0	4.0
WIENER	75	103	124	141	167	189	206	242	269	309	338	378	425	472	488	494
MD1 QUEUE	59	87	108	125	153	175	193	229	257	298	328	370	419	469	486	493
ME1 K= 4.00	56	84	105	123	150	172	190	226	254	296	326	368	418	468	486	493
ME1 K= 3.00	55	84	105	122	149	171	190	226	254	295	326	368	417	468	486	493
ME1 K= 2.00	54	83	104	121	148	170	189	225	253	294	325	367	417	468	486	493
ME1 K= 1.50	53	82	103	120	147	169	188	224	252	294	324	367	416	467	485	493
ME1 K= 1.00	52	81	102	119	146	168	186	223	251	293	323	366	416	467	485	493
ME1 K= 0.50	51	79	100	117	144	166	184	221	249	291	322	365	415	467	485	493
ME1 K= 0.20	49	77	98	115	142	164	182	219	247	289	320	363	414	466	485	493
GAMMA INPUT	48	75	96	113	140	162	180	217	245	288	319	362	413	466	484	492

RHO = 0.80 SCALED INITIAL SERVER LOAD = 4.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH 0.40	0.60	0.80	1.00	1.50	2.00	2.50	3.00	4.00	5.0	6.0	7.0	8.0	10.0	12.0	14.0
WIENER	3600	3400	3200	3001	2514	2073	1701	1404	1001	774	649	582	545	514	504	501
MD1 QUEUE	3600	3400	3200	3000	2508	2058	1682	1386	989	769	649	583	547	515	505	502
ME1 K= 4.00	3600	3400	3200	3000	2507	2055	1678	1382	987	768	649	583	547	515	505	502
ME1 K= 3.00	3600	3400	3200	3000	2506	2054	1677	1381	986	768	649	583	547	515	505	502
ME1 K= 2.00	3600	3400	3200	3000	2506	2053	1676	1379	985	768	649	583	547	515	505	502
ME1 K= 1.50	3600	3400	3200	3000	2506	2052	1674	1378	985	767	649	583	547	515	505	502
ME1 K= 1.00	3600	3400	3200	3000	2505	2051	1672	1376	983	767	649	583	547	516	505	502
ME1 K= 0.50	3600	3400	3200	3000	2505	2049	1669	1373	981	766	648	584	548	516	505	502
ME1 K= 0.20	3600	3400	3200	3000	2504	2046	1666	1370	979	765	648	584	548	516	505	502
GAMMA INPUT	3600	3400	3200	3000	2503	2044	1663	1367	977	764	648	584	548	516	506	502

RHO = 0.80 SCALED INITIAL SERVER LOAD = 3.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH 0.40	0.60	0.80	1.00	1.50	2.00	2.50	3.00	4.00	5.0	6.0	7.0	8.0	10.0	12.0	14.0
WIENER	2600	2400	2203	2013	1596	1278	1049	887	694	599	551	527	514	504	501	500
MD1 QUEUE	2600	2400	2201	2006	1578	1258	1032	875	689	598	552	528	515	505	502	501
ME1 K= 4.00	2600	2400	2200	2005	1574	1253	1028	872	688	598	552	528	516	505	502	501
ME1 K= 3.00	2600	2400	2205	2005	1574	1252	1027	871	688	598	552	528	516	505	502	501
ME1 K= 2.00	2600	2400	2200	2004	1572	1251	1026	870	688	598	552	528	516	505	502	501
ME1 K= 1.50	2600	2400	2200	2004	1571	1249	1024	869	687	598	552	528	516	505	502	501
ME1 K= 1.00	2600	2400	2200	2004	1569	1247	1023	868	687	598	552	528	516	505	502	501
ME1 K= 0.50	2600	2400	2200	2003	1567	1244	1020	866	686	597	552	529	516	505	502	501
ME1 K= 0.20	2500	2400	2200	2003	1564	1241	1017	863	685	597	552	529	516	505	502	501
GAMMA INPUT	2600	2400	2200	2002	1561	1237	1014	861	684	597	552	529	516	505	502	501

RHO = 0.80 SCALED INITIAL SERVER LOAD = 2.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH 0.40	0.60	0.80	1.00	1.50	2.00	2.50	3.00	4.00	5.0	6.0	7.0	8.0	10.0	12.0	14.0
WIENER	1602	1418	1258	1126	891	750	664	609	550	524	512	506	503	501	500	500
MD1 QUEUE	1600	1407	1240	1104	870	735	654	603	549	524	512	506	503	501	500	500
ME1 K= 4.00	1600	1406	1237	1100	866	732	652	602	548	524	512	507	504	501	500	500
ME1 K= 3.00	1600	1406	1236	1098	865	732	651	601	548	524	512	507	504	501	500	500
ME1 K= 2.00	1600	1405	1235	1097	864	730	651	601	548	524	512	507	504	501	500	500
ME1 K= 1.50	1600	1405	1234	1095	862	729	650	600	548	524	512	507	504	501	500	500
ME1 K= 1.00	1600	1404	1232	1093	860	728	649	600	547	524	512	507	504	501	500	500
ME1 K= 0.50	1600	1403	1230	1090	857	725	647	599	547	524	512	507	504	501	500	500
ME1 K= 0.20	1600	1403	1228	1087	853	723	645	598	547	523	512	507	504	501	500	500
GAMMA INPUT	1600	1402	1226	1084	850	720	644	596	546	523	512	507	504	501	500	500

RHO = 0.80 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.50	0.60	0.8	1.0	2.0	3.0	4.0	5.0	6.0
WIENER	900	853	812	775	744	717	694	657	629	590	565	517	506	502	501	500
MD1 QUEUE	900	850	800	758	723	693	669	631	603	567	546	509	502	501	500	500
ME1 K= 4.00	900	850	800	756	719	689	664	626	598	563	542	508	502	500	500	500
ME1 K= 3.00	900	850	800	756	719	688	663	625	597	562	541	507	501	500	500	500
ME1 K= 2.00	900	850	800	755	717	686	661	622	595	560	540	506	501	500	500	500
ME1 K= 1.50	900	850	800	754	716	685	660	621	594	559	538	506	501	500	500	500
ME1 K= 1.00	900	850	800	754	715	683	657	618	591	556	536	505	500	500	500	500
ME1 K= 0.50	900	850	800	753	713	680	654	615	587	553	533	504	500	499	499	500
ME1 K= 0.20	900	850	800	752	711	678	651	611	583	549	530	502	499	499	499	499
GAMMA INPUT	900	850	800	751	709	675	648	607	579	546	527	501	498	498	499	499

RHO = 0.80 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.50	0.60	0.8	1.0	2.0	3.0	4.0	5.0	6.0
WIENER	703	663	632	608	589	574	561	543	531	516	508	498	498	499	499	500
MD1 QUEUE	700	650	612	583	562	546	534	517	507	495	491	491	495	497	498	499
ME1 K= 4.00	700	650	609	579	557	541	529	512	502	491	487	489	494	497	498	499
ME1 K= 3.00	700	650	608	578	556	540	528	511	501	490	486	489	494	496	498	499
ME1 K= 2.00	700	650	607	577	554	538	525	509	499	489	485	488	493	496	498	499
ME1 K= 1.50	700	650	606	575	553	536	524	507	497	487	484	487	493	496	498	499
ME1 K= 1.00	700	650	606	574	551	534	521	505	495	485	482	487	493	496	498	499
ME1 K= 0.50	700	650	604	571	547	530	517	501	491	482	479	485	492	496	498	499
ME1 K= 0.20	700	650	603	568	544	526	513	497	487	479	476	484	491	495	497	499
GAMMA INPUT	700	650	602	566	541	523	510	493	484	475	474	483	490	495	497	498

RHO = 0.80 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.50	0.60	0.8	1.0	2.0	3.0	4.0	5.0	6.0
WIENER	514	491	477	468	463	460	458	457	458	462	467	485	493	496	498	499
MD1 QUEUE	500	467	449	439	435	432	432	433	436	444	452	478	489	495	497	499
ME1 K= 4.00	500	463	444	434	429	427	427	428	432	441	449	476	489	494	497	498
ME1 K= 3.00	500	462	443	433	428	426	425	427	431	440	448	476	488	494	497	498
ME1 K= 2.00	500	460	441	431	426	424	423	425	429	438	447	475	488	494	497	498
ME1 K= 1.50	500	460	439	429	424	422	422	424	428	437	446	475	488	494	497	498
ME1 K= 1.00	500	458	437	427	422	419	419	421	426	435	444	474	487	494	497	498
ME1 K= 0.50	500	456	434	423	417	415	415	418	422	432	442	473	487	493	496	498
ME1 K= 0.20	500	455	431	419	413	411	411	414	419	429	439	472	486	493	496	498
GAMMA INPUT	500	454	428	416	410	407	407	410	415	426	437	470	485	492	496	498

RHO = 0.80 SCALED INITIAL SERVER LOAD = 0.40 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.30	0.40	0.6	0.8	1.0	1.5	2.0	3.0	4.0
WIENER	380	365	357	353	351	354	360	374	388	411	428	442	464	477	490	495
MD1 QUEUE	380	360	340	320	324	323	329	347	363	391	412	428	454	470	486	493
ME1 K= 4.00	380	360	340	320	318	317	324	342	359	387	408	425	452	469	485	493
ME1 K= 3.00	380	360	340	320	316	316	323	340	358	386	408	424	452	468	485	493
ME1 K= 2.00	380	360	340	320	315	314	320	338	356	384	406	423	451	468	485	492
ME1 K= 1.50	380	360	340	320	314	312	319	337	354	383	405	422	450	467	485	492
ME1 K= 1.00	380	360	340	320	312	310	316	334	352	381	403	421	449	466	484	492
ME1 K= 0.50	380	360	340	320	310	306	312	330	348	378	401	418	448	465	483	492
ME1 K= 0.20	380	360	340	320	308	302	308	326	344	375	398	416	446	464	483	491
GAMMA INPUT	380	360	340	320	306	299	304	322	341	372	395	414	444	463	482	491

RHO = 0.80 SCALED INITIAL SERVER LOAD = 0.20 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.30	0.40	0.6	0.8	1.0	1.5	2.0	3.0	4.0
WIENER	193	204	218	231	243	269	290	324	349	385	410	428	457	473	488	494
MD1 QUEUE	180	160	185	197	209	238	262	299	327	367	395	415	448	466	485	492
ME1 K= 4.00	180	160	178	192	205	233	257	294	322	363	392	412	446	465	484	492
ME1 K= 3.00	180	160	177	190	203	232	256	293	321	362	391	412	445	464	484	492
ME1 K= 2.00	180	160	175	188	201	230	253	291	320	361	390	411	444	464	483	492
ME1 K= 1.50	180	160	173	186	199	228	252	289	318	360	389	410	444	463	483	492
ME1 K= 1.00	180	160	171	184	197	225	249	287	316	358	387	408	443	463	483	491
ME1 K= 0.50	180	160	168	180	193	221	245	284	313	355	384	406	441	461	482	491
ME1 K= 0.20	180	160	165	177	189	218	242	280	309	352	382	404	440	460	481	490
GAMMA INPUT	180	160	163	174	186	214	238	276	306	349	379	402	438	459	480	490

RHO = 0.80 SCALED INITIAL SERVER LOAD = 0.0 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.30	0.40	0.6	0.8	1.0	1.5	2.0	3.0	4.0
WIENER	103	141	167	189	206	242	269	309	338	378	405	425	455	472	488	494
MD1 QUEUE	65	108	134	157	176	214	242	285	316	360	390	412	446	465	484	492
ME1 K= 4.00	63	102	130	152	171	208	237	280	312	357	387	409	444	464	483	492
ME1 K= 3.00	62	100	128	151	170	207	236	279	311	356	386	408	443	463	483	492
ME1 K= 2.00	61	99	126	149	168	205	234	277	309	354	385	407	443	463	483	491
ME1 K= 1.50	60	97	125	147	166	203	232	276	308	353	384	406	442	462	482	491
ME1 K= 1.00	58	95	123	145	164	201	230	274	306	351	382	405	441	461	482	491
ME1 K= 0.50	57	92	119	141	160	197	226	270	303	348	380	403	439	460	481	491
ME1 K= 0.20	55	89	116	138	156	194	223	267	299	345	377	400	438	459	481	490
GAMMA INPUT	53	87	113	135	153	190	219	263	296	343	375	398	436	458	480	490

RHO = 0.70 SCALED INITIAL SERVER LOAD = 4.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.50	1.00	1.50	2.00	2.50	3.00	4.00	5.00	6.00	7.0	8.0	9.0	10.0	12.0	14.0	16.0
WIENER	3500	3001	2514	2073	1701	1404	1001	774	649	582	545	525	514	504	501	500
MD1 QUEUE	3500	3000	2504	2047	1668	1372	981	766	649	584	548	527	516	505	502	501
ME1 K= 4.00	3500	3000	2503	2042	1661	1365	977	765	648	584	548	528	516	506	502	501
ME1 K= 3.00	3500	3000	2502	2041	1660	1364	976	764	648	584	548	528	516	506	502	501
ME1 K= 2.00	3500	3000	2502	2039	1657	1361	974	763	648	584	548	528	516	506	502	501
ME1 K= 1.50	3500	3000	2502	2038	1655	1359	973	763	648	584	549	528	517	506	502	501
ME1 K= 1.00	3500	3000	2501	2036	1652	1356	971	762	648	584	549	529	517	506	502	501
ME1 K= 0.50	3500	3000	2501	2033	1646	1350	967	760	647	585	549	529	517	506	502	501
ME1 K= 0.20	3500	3000	2501	2030	1641	1345	964	759	647	585	549	529	517	506	502	501
GAMMA INPUT	3500	3000	2500	2027	1636	1339	960	757	646	585	550	530	518	507	502	501

RHO = 0.70 SCALED INITIAL SERVER LOAD = 3.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.50	1.00	1.50	2.00	2.50	3.00	4.00	5.00	6.00	7.0	8.0	9.0	10.0	12.0	14.0	16.0
WIENER	2500	2013	1596	1278	1049	887	694	599	551	527	514	508	504	501	500	500
MD1 QUEUE	2500	2002	1565	1243	1019	866	686	597	552	529	516	509	505	502	501	500
ME1 K= 4.00	2500	2001	1559	1235	1013	861	684	597	552	529	516	509	505	502	501	500
ME1 K= 3.00	2500	2001	1558	1234	1011	860	684	597	552	529	516	509	505	502	501	500
ME1 K= 2.00	2500	2001	1555	1231	1009	858	683	597	552	529	516	509	505	502	501	500
ME1 K= 1.50	2500	2001	1554	1228	1007	856	682	596	552	529	516	509	505	502	501	500
ME1 K= 1.00	2500	2000	1551	1225	1004	854	681	596	552	529	517	510	506	502	501	500
ME1 K= 0.50	2500	2000	1547	1219	999	850	679	595	552	529	517	510	506	502	501	500
ME1 K= 0.20	2500	2000	1543	1214	993	846	677	595	552	529	517	510	506	502	501	500
GAMMA INPUT	2500	2000	1539	1208	988	842	675	594	552	530	517	510	506	502	501	500

RHO = 0.70 SCALED INITIAL SERVER LOAD = 2.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.50	1.00	1.50	2.00	2.50	3.00	4.00	5.00	6.00	7.0	8.0	9.0	10.0	12.0	14.0	16.0
WIENER	1507	1126	891	750	664	609	550	524	512	506	503	502	501	500	500	500
MD1 QUEUE	1500	1088	856	725	647	599	547	524	512	507	504	502	501	500	500	500
ME1 K= 4.00	1500	1081	849	720	644	596	546	523	512	507	504	502	501	500	500	500
ME1 K= 3.00	1500	1079	847	718	643	596	546	523	512	507	504	502	501	500	500	500
ME1 K= 2.00	1500	1076	844	716	641	595	546	523	512	507	504	502	501	500	500	500
ME1 K= 1.50	1500	1074	841	714	640	594	545	523	512	507	504	502	501	500	500	500
ME1 K= 1.00	1500	1071	838	712	638	593	545	523	512	507	504	502	501	500	500	500
ME1 K= 0.50	1500	1066	832	707	635	591	544	522	512	507	504	502	501	500	500	500
ME1 K= 0.20	1500	1061	826	703	632	589	543	522	512	507	504	502	501	500	500	500
GAMMA INPUT	1500	1056	821	698	629	587	542	522	512	506	504	502	501	500	500	500

RHO = 0.70 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.15	0.20	0.25	0.30	0.40	0.50	0.60	0.80	1.00	1.5	2.0	3.0	4.0	5.0	7.0	9.0
WIENER	853	812	775	744	694	657	629	590	565	532	517	506	502	501	500	500
MD1 QUEUE	850	800	750	700	650	612	585	552	533	511	504	500	499	499	500	500
ME1 K= 4.00	850	800	750	700	642	603	577	544	526	507	501	498	498	499	500	500
ME1 K= 3.00	850	800	750	700	641	601	575	542	525	506	500	498	498	499	500	500
ME1 K= 2.00	850	800	750	700	638	598	571	539	522	504	499	497	498	499	499	500
ME1 K= 1.50	850	800	750	700	636	595	569	537	520	502	498	497	498	498	499	500
ME1 K= 1.00	850	800	750	700	633	592	565	533	517	500	496	496	497	498	499	500
ME1 K= 0.50	850	800	750	700	628	586	558	527	511	497	494	494	496	498	499	500
ME1 K= 0.20	850	800	750	700	624	580	552	521	506	493	491	493	496	497	499	500
GAMMA INPUT	850	800	750	700	621	575	546	516	501	489	488	492	495	497	499	500

RHO = 0.70 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.15	0.20	0.25	0.30	0.40	0.50	0.60	0.80	1.00	1.5	2.0	3.0	4.0	5.0	7.0	9.0
WIENER	663	632	608	589	561	543	531	516	508	500	498	498	499	499	500	500
MD1 QUEUE	650	600	559	546	514	499	489	481	479	481	485	492	496	498	499	500
ME1 K= 4.00	650	600	556	535	506	490	482	474	473	477	483	490	495	497	499	500
ME1 K= 3.00	650	600	555	533	504	488	480	473	472	476	482	490	495	497	499	500
ME1 K= 2.00	650	600	554	530	500	485	476	470	469	474	481	489	494	497	499	500
ME1 K= 1.50	650	600	554	528	497	482	474	468	467	473	480	489	494	497	499	500
ME1 K= 1.00	650	600	553	525	494	478	470	464	464	471	478	488	494	496	499	500
ME1 K= 0.50	650	600	552	521	488	472	464	459	460	468	476	487	493	496	499	500
ME1 K= 0.20	650	600	551	517	482	466	458	453	455	464	474	486	492	495	498	499
GAMMA INPUT	650	600	551	514	476	460	452	448	450	461	471	484	491	495	498	499

RHO = 0.70 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.15	0.20	0.25	0.30	0.40	0.50	0.60	0.80	1.00	1.5	2.0	3.0	4.0	5.0	7.0	9.0
WIENER	491	477	468	463	458	457	458	462	467	478	485	493	496	498	500	500
MD1 QUEUE	450	423	423	411	414	416	421	432	441	460	473	487	493	496	499	500
ME1 K= 4.00	450	418	410	405	404	408	414	425	436	457	470	485	492	496	499	500
ME1 K= 3.00	450	417	408	403	402	406	412	424	435	456	470	485	492	496	499	500
ME1 K= 2.00	450	415	404	400	399	403	409	421	433	454	469	484	492	496	499	500
ME1 K= 1.50	450	413	402	397	396	401	407	419	431	453	468	484	491	495	499	500
ME1 K= 1.00	450	411	398	393	392	397	403	416	428	451	466	483	491	495	498	499
ME1 K= 0.50	450	408	393	387	386	391	397	411	424	448	464	482	490	495	498	499
ME1 K= 0.20	450	406	388	381	380	385	392	406	420	445	462	480	489	494	498	499
GAMMA INPUT	450	404	384	376	374	379	386	402	416	442	459	479	489	494	498	499

RHO = 0.70 SCALED INITIAL SERVER LOAD = 0.40 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.30	0.40	0.6	0.8	1.0	1.5	2.0	3.0	4.0
WIENER	380	365	357	353	351	354	360	374	388	411	428	442	464	477	490	495
MD1 QUEUE	380	360	340	320	300	298	313	329	346	377	400	418	448	465	483	492
ME1 K= 4.00	380	360	340	320	300	290	299	319	339	371	395	413	444	463	482	491
ME1 K= 3.00	380	360	340	320	300	288	297	317	337	369	393	412	443	462	482	491
ME1 K= 2.00	380	360	340	320	300	285	294	314	334	366	391	410	442	461	481	490
ME1 K= 1.50	380	360	340	320	300	283	291	311	331	364	389	408	441	460	481	490
ME1 K= 1.00	380	360	340	320	300	280	287	308	328	361	386	406	439	459	480	490
ME1 K= 0.50	380	360	340	320	300	276	281	301	322	356	382	402	436	457	478	489
ME1 K= 0.20	380	360	340	320	300	272	276	296	316	351	377	398	433	454	477	488
GAMMA INPUT	380	360	340	320	300	269	271	290	311	346	373	394	430	452	476	487

RHO = 0.70 SCALED INITIAL SERVER LOAD = 0.20 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.30	0.40	0.6	0.8	1.0	1.5	2.0	3.0	4.0
WIENER	193	204	218	231	243	269	290	324	349	385	410	428	457	473	488	494
MD1 QUEUE	180	160	140	166	186	221	240	282	312	354	384	406	441	461	482	491
ME1 K= 4.00	180	160	140	161	178	209	234	274	304	348	379	401	438	459	480	490
ME1 K= 3.00	180	160	140	160	176	207	233	272	303	347	377	400	437	458	480	490
ME1 K= 2.00	180	160	140	158	173	204	229	269	300	344	375	398	435	457	479	489
ME1 K= 1.50	180	160	140	157	171	201	227	267	297	342	373	397	434	456	479	489
ME1 K= 1.00	180	160	140	154	168	198	223	263	294	339	371	394	432	455	478	489
ME1 K= 0.50	180	160	140	151	163	192	217	258	289	334	366	390	430	453	477	488
ME1 K= 0.20	180	160	140	148	159	188	212	252	283	330	362	387	427	450	475	487
GAMMA INPUT	180	160	140	145	156	183	207	247	278	325	358	383	424	448	474	486

RHO = 0.70 SCALED INITIAL SERVER LOAD = 0.0 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.30	0.40	0.6	0.8	1.0	1.5	2.0	3.0	4.0
WIENER	103	141	167	189	206	242	269	309	338	378	405	425	455	472	488	494
MD1 QUEUE	43	80	111	136	157	194	224	268	301	348	379	402	439	460	481	491
ME1 K= 4.00	42	77	105	128	148	186	216	261	294	342	374	398	436	458	480	490
ME1 K= 3.00	42	76	104	126	146	184	214	259	293	340	373	397	435	457	480	490
ME1 K= 2.00	42	75	101	124	143	181	211	256	290	338	371	395	434	456	479	489
ME1 K= 1.50	41	74	100	122	141	179	209	254	288	336	369	393	432	455	478	489
ME1 K= 1.00	41	72	98	119	138	176	205	251	285	333	366	391	431	454	478	488
ME1 K= 0.50	39	70	94	115	133	171	200	245	279	328	362	387	428	452	476	488
ME1 K= 0.20	38	67	91	111	129	166	195	240	274	323	358	383	425	449	475	487
GAMMA INPUT	37	65	88	108	125	161	190	235	269	319	354	379	422	447	474	486

RHO = 0.60 SCALED INITIAL SERVER LOAD = 4.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	5.00	6.0	7.0	8.0	10.0	12.0	15.0	20.0
WIENER	3500	3001	2514	2073	1701	1404	1175	1001	774	649	582	545	514	504	501	500
MD1 QUEUE	3500	3000	2500	2032	1648	1354	1133	971	762	648	585	549	517	506	501	500
ME1 K= 4.00	3500	3000	2500	2026	1638	1343	1125	964	759	647	585	550	517	506	501	500
ME1 K= 3.00	3500	3000	2500	2025	1636	1341	1123	963	759	647	585	550	518	506	502	500
ME1 K= 2.00	3500	3000	2500	2022	1632	1337	1119	960	757	647	585	550	518	507	502	500
ME1 K= 1.50	3500	3000	2500	2021	1629	1333	1116	958	756	646	585	550	518	507	502	500
ME1 K= 1.00	3500	3000	2500	2018	1624	1328	1112	954	755	646	585	550	518	507	502	500
ME1 K= 0.50	3500	3000	2500	2015	1616	1320	1104	949	752	645	585	551	519	507	502	500
ME1 K= 0.20	3500	3000	2500	2012	1609	1311	1097	943	749	644	585	551	519	507	502	500
GAMMA INPUT	3500	3000	2500	2009	1602	1303	1090	937	746	643	585	551	519	508	502	500

RHO = 0.60 SCALED INITIAL SERVER LOAD = 3.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	5.00	6.0	7.0	8.0	10.0	12.0	15.0	20.0
WIENER	2500	2013	1596	1278	1049	887	774	694	599	551	527	514	504	501	500	500
MD1 QUEUE	2500	2000	1547	1222	1002	854	752	681	596	553	529	517	506	502	500	500
ME1 K= 4.00	2500	2000	1538	1211	993	846	747	678	595	552	530	517	506	502	500	500
ME1 K= 3.00	2500	2000	1536	1208	990	845	746	677	595	552	530	517	506	502	500	500
ME1 K= 2.00	2500	2000	1533	1204	986	842	743	676	594	552	530	517	506	502	501	500
ME1 K= 1.50	2500	2000	1531	1200	983	839	742	675	594	552	530	517	506	502	501	500
ME1 K= 1.00	2500	2000	1528	1195	978	835	739	673	593	552	530	517	506	502	501	500
ME1 K= 0.50	2500	2000	1523	1186	970	829	735	670	592	552	530	518	506	502	501	500
ME1 K= 0.20	2500	2000	1519	1178	962	823	730	667	591	551	530	518	507	503	501	500
GAMMA INPUT	2500	2000	1515	1170	954	817	726	663	589	551	530	518	507	503	501	500

RHO = 0.60 SCALED INITIAL SERVER LOAD = 2.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	5.00	6.0	7.0	8.0	10.0	12.0	15.0	20.0
WIENER	1507	1126	891	750	664	609	574	550	524	512	506	503	501	500	500	500
MD1 QUEUE	1500	1066	836	711	638	593	564	545	523	512	507	504	501	500	500	500
ME1 K= 4.00	1500	1055	825	703	632	589	562	543	522	512	507	504	501	500	500	500
ME1 K= 3.00	1500	1053	822	701	631	588	561	543	522	512	507	504	501	500	500	500
ME1 K= 2.00	1500	1049	817	697	629	587	560	542	522	512	507	504	501	500	500	500
ME1 K= 1.50	1500	1046	814	694	627	585	559	542	522	512	507	504	501	500	500	500
ME1 K= 1.00	1500	1041	808	690	624	583	558	541	521	511	506	504	501	500	500	500
ME1 K= 0.50	1500	1035	799	683	619	580	555	539	520	511	506	504	501	500	500	500
ME1 K= 0.20	1500	1029	791	676	614	576	553	537	519	511	506	504	501	500	500	500
GAMMA INPUT	1500	1025	783	670	609	573	550	535	519	510	506	503	501	500	500	500

RHO = 0.60 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.20	0.30	0.40	0.50	0.60	0.80	1.00	1.50	2.00	2.5	3.0	4.0	5.0	6.0	8.0	10.0
WIENER	812	744	694	657	629	590	565	532	517	510	506	502	501	500	500	500
MD1 QUEUE	800	700	600	592	561	533	515	500	496	495	496	497	498	499	500	500
ME1 K= 4.00	800	700	600	573	548	519	505	493	491	492	493	496	497	498	499	500
ME1 K= 3.00	800	700	600	570	544	517	503	491	490	491	493	495	497	498	499	500
ME1 K= 2.00	800	700	600	565	539	512	499	489	488	490	492	495	497	498	499	500
ME1 K= 1.50	800	700	600	561	535	508	496	486	486	489	491	494	497	498	499	500
ME1 K= 1.00	800	700	600	556	530	503	491	483	484	487	489	494	496	498	499	500
ME1 K= 0.50	800	700	600	548	521	494	483	477	480	484	487	492	495	497	499	500
ME1 K= 0.20	800	700	600	542	513	486	475	471	476	480	485	491	494	497	499	499
GAMMA INPUT	800	700	600	536	506	478	468	466	471	477	482	489	493	496	498	499

RHO = 0.60 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.20	0.30	0.40	0.50	0.60	0.80	1.00	1.50	2.00	2.5	3.0	4.0	5.0	6.0	8.0	10.0
WIENER	632	589	561	543	531	516	508	500	498	498	498	499	499	500	500	500
MD1 QUEUE	600	500	489	479	462	464	463	471	478	484	488	494	496	498	499	500
ME1 K= 4.00	600	500	474	461	455	452	455	465	474	481	486	492	496	497	499	500
ME1 K= 3.00	600	500	471	458	452	450	452	463	473	480	485	492	495	497	499	500
ME1 K= 2.00	600	500	465	453	447	445	449	461	471	478	484	491	495	497	499	500
ME1 K= 1.50	600	500	462	449	443	442	446	458	469	477	483	491	495	497	499	500
ME1 K= 1.00	600	500	457	443	438	437	441	455	467	476	482	490	494	497	499	500
ME1 K= 0.50	600	500	449	435	429	429	434	450	463	473	480	488	493	496	499	499
ME1 K= 0.20	600	500	443	427	421	421	427	445	459	470	477	487	492	495	498	499
GAMMA INPUT	600	500	437	419	413	414	421	440	455	466	475	485	491	495	498	499

RHO = 0.60 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

	SCALED EPOCH															
	0.20	0.30	0.40	0.50	0.60	0.80	1.00	1.50	2.00	2.5	3.0	4.0	5.0	6.0	8.0	10.0
WIENER	477	463	458	457	458	462	467	478	485	490	493	496	498	499	500	500
MD1 QUEUE	400	382	396	389	402	414	428	451	466	476	483	491	495	497	499	500
ME1 K= 4.00	400	371	376	382	390	406	420	445	462	473	481	490	494	497	499	500
ME1 K= 3.00	400	368	373	379	387	403	418	444	461	472	480	489	494	497	499	500
ME1 K= 2.00	400	363	367	375	383	400	414	441	459	471	479	489	494	496	499	500
ME1 K= 1.50	400	360	363	371	380	397	411	439	458	470	478	488	493	496	499	500
ME1 K= 1.00	400	355	358	366	374	392	408	436	455	468	477	487	493	496	499	499
ME1 K= 0.50	400	348	350	357	366	385	401	432	452	465	474	486	492	495	498	499
ME1 K= 0.20	400	343	342	349	359	378	395	427	448	462	472	484	491	495	498	499
GAMMA INPUT	400	337	335	342	351	371	388	422	444	459	470	483	490	494	498	499

RHO = 0.60	SCALED INITIAL SERVER LOAD = 0.40						SCALED EPOCH			SCALED MEAN WAIT IN THOUSANDTHS						
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.2	1.5	2.0	3.0	4.0	5.0
WIENER	360	351	354	360	374	388	400	411	428	442	452	464	477	490	495	498
MD1 QUEUE	350	300	250	267	310	323	342	361	385	405	421	439	459	480	489	494
ME1 K= 4.00	350	300	250	260	291	313	333	349	377	398	414	433	454	477	488	493
ME1 K= 3.00	350	300	250	259	288	310	330	347	375	396	412	432	453	477	488	493
ME1 K= 2.00	350	300	250	255	283	306	326	343	371	393	410	429	452	476	487	493
ME1 K= 1.50	350	300	250	253	279	302	322	340	368	390	408	427	450	475	486	492
ME1 K= 1.00	350	300	250	249	274	297	318	335	364	386	404	425	448	474	486	492
ME1 K= 0.50	350	300	250	244	266	289	310	328	357	380	399	420	444	471	484	491
ME1 K= 0.20	350	300	250	239	260	282	303	321	351	374	393	415	441	469	483	490
GAMMA INPUT	350	300	250	235	253	275	296	314	345	369	388	411	437	467	481	489

RHO = 0.60	SCALED INITIAL SERVER LOAD = 0.20						SCALED EPOCH			SCALED MEAN WAIT IN THOUSANDTHS						
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.2	1.5	2.0	3.0	4.0	5.0
WIENER	211	243	269	290	324	349	369	385	410	428	442	457	473	488	494	497
MD1 QUEUE	150	141	185	220	262	290	318	337	370	393	412	432	455	478	489	494
ME1 K= 4.00	150	139	177	206	249	281	307	328	362	386	405	427	451	476	487	493
ME1 K= 3.00	150	138	175	203	246	279	305	326	360	384	404	425	450	475	487	493
ME1 K= 2.00	150	137	172	199	242	275	301	322	356	381	401	423	448	474	486	492
ME1 K= 1.50	150	136	169	196	239	271	298	319	354	379	399	421	446	473	486	492
ME1 K= 1.00	150	134	166	192	234	267	293	315	350	376	396	419	444	472	485	492
ME1 K= 0.50	150	131	160	185	227	259	286	308	[illegible]	370	390	414	441	470	483	491
ME1 K= 0.20	150	129	155	180	220	253	279	302	337	364	385	409	437	467	482	490
GAMMA INPUT	150	126	151	174	214	246	273	295	331	358	380	405	433	465	480	489

RHO = 0.60	SCALED INITIAL SERVER LOAD = 0.0						SCALED EPOCH			SCALED MEAN WAIT IN THOUSANDTHS						
	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.2	1.5	2.0	3.0	4.0	5.0
WIENER	155	206	242	269	309	338	360	378	405	425	439	455	472	488	494	497
MD1 QUEUE	68	124	168	203	247	282	310	331	365	390	409	430	454	478	488	494
ME1 K= 4.00	67	118	158	190	237	272	300	322	357	383	403	425	449	475	487	493
ME1 K= 3.00	66	117	156	187	234	270	297	320	355	381	401	424	448	475	487	493
ME1 K= 2.00	65	114	153	183	230	266	294	316	352	378	399	421	447	474	486	492
ME1 K= 1.50	64	113	150	180	227	263	291	314	349	376	396	419	445	473	485	492
ME1 K= 1.00	63	110	147	176	223	258	286	309	345	372	393	417	443	471	485	491
ME1 K= 0.50	61	106	141	170	216	251	279	303	339	367	388	412	440	469	483	490
ME1 K= 0.20	60	102	137	165	210	245	273	296	333	361	383	408	436	467	482	490
GAMMA INPUT	58	99	132	160	204	239	267	290	327	356	378	403	432	464	480	488

RHO = 0.50 SCALED INITIAL SERVER LOAD = 4.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.0	6.0	7.0	8.0	10.0	15.0	20.0
WIENER	3500	3001	2514	2073	1701	1404	1175	1001	870	774	649	582	545	514	501	500
MD1 QUEUE	3500	3000	2500	2000	1621	1327	1113	956	841	756	647	586	551	518	502	500
ME1 K= 4.00	3500	3000	2500	2000	1606	1312	1100	946	834	752	645	586	551	519	502	500
ME1 K= 3.00	3500	3000	2500	2000	1602	1308	1096	943	832	750	645	586	551	519	502	500
ME1 K= 2.00	3500	3000	2500	2000	1596	1302	1091	939	829	748	644	586	552	520	502	500
ME1 K= 1.50	3500	3000	2500	2000	1591	1296	1086	936	826	747	644	585	552	520	502	500
ME1 K= 1.00	3500	3000	2500	2000	1584	1289	1080	930	823	744	643	585	552	520	502	500
ME1 K= 0.50	3500	3000	2500	2000	1574	1276	1068	922	816	739	641	585	552	520	502	500
ME1 K= 0.20	3500	3000	2500	2000	1565	1264	1058	913	810	735	639	584	552	521	503	500
GAMMA INPUT	3500	3000	2500	2000	1557	1253	1047	904	803	730	636	583	552	521	503	500

RHO = 0.50 SCALED INITIAL SERVER LOAD = 3.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.0	6.0	7.0	8.0	10.0	15.0	20.0
WIENER	2500	2013	1596	1278	1049	887	774	694	639	599	551	527	514	504	500	500
MD1 QUEUE	2500	2000	1500	1192	980	837	741	675	628	594	553	530	518	506	501	500
ME1 K= 4.00	2500	2000	1500	1176	965	826	733	669	624	592	552	530	518	506	501	500
ME1 K= 3.00	2500	2000	1500	1171	961	823	731	668	623	592	552	530	518	507	501	500
ME1 K= 2.00	2500	2000	1500	1165	955	819	728	666	622	591	552	530	518	507	501	500
ME1 K= 1.50	2500	2000	1500	1159	950	815	725	664	621	590	551	530	518	507	501	500
ME1 K= 1.00	2500	2000	1500	1152	942	809	721	661	619	589	551	530	518	507	501	500
ME1 K= 0.50	2500	2000	1500	1140	930	799	714	656	615	586	550	530	518	507	501	500
ME1 K= 0.20	2500	2000	1500	1128	919	790	707	651	612	584	549	529	518	507	501	500
GAMMA INPUT	2500	2000	1500	1118	908	781	700	645	608	581	547	529	518	507	501	500

RHO = 0.50 SCALED INITIAL SERVER LOAD = 2.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.0	6.0	7.0	8.0	10.0	15.0	20.0
WIENER	1507	1126	891	750	664	609	574	550	535	524	512	506	503	501	500	500
MD1 QUEUE	1500	1000	804	693	625	585	559	542	530	522	512	507	504	501	500	500
ME1 K= 4.00	1500	1000	791	680	617	579	555	539	528	520	511	506	504	501	500	500
ME1 K= 3.00	1500	1000	787	676	615	577	554	538	527	520	511	506	504	501	500	500
ME1 K= 2.00	1500	1000	780	671	611	575	552	537	526	519	511	506	504	501	500	500
ME1 K= 1.50	1500	1000	775	667	608	573	550	536	526	519	510	505	503	501	500	500
ME1 K= 1.00	1500	1000	767	660	603	569	548	534	524	518	510	506	503	501	500	500
ME1 K= 0.50	1500	1001	754	650	596	564	544	531	522	516	509	505	503	501	500	500
ME1 K= 0.20	1500	1000	743	640	588	558	540	528	520	514	508	505	503	501	500	500
GAMMA INPUT	1500	1000	731	630	580	552	535	525	517	513	507	504	502	501	500	500

RHO = 0.50 SCALED INITIAL SERVER LOAD = 1.00 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH	0.30	0.40	0.50	0.60	0.80	1.00	1.50	2.00	2.50	3.0	4.0	5.0	6.0	7.0	8.0	10.0
WIENER	744	694	657	629	590	565	532	517	510	506	502	501	500	500	500	500
MD1 QUEUE	700	600	500	515	515	484	483	486	488	490	494	496	498	499	499	500
ME1 K= 4.00	700	600	500	501	486	477	474	478	482	486	492	495	497	498	499	500
ME1 K= 3.00	700	600	500	497	481	473	472	476	481	485	491	495	497	498	499	500
ME1 K= 2.00	700	600	500	491	474	467	467	473	479	483	490	494	496	498	499	499
ME1 K= 1.50	700	600	500	486	469	463	464	470	477	482	489	493	496	497	498	499
ME1 K= 1.00	700	600	500	480	461	456	459	466	474	480	488	492	495	497	498	499
ME1 K= 0.50	700	600	500	470	450	445	450	460	469	476	485	491	494	496	498	499
ME1 K= 0.20	700	600	500	461	439	434	442	454	464	472	483	489	493	496	497	499
GAMMA INPUT	700	600	500	454	429	425	434	447	459	468	480	487	492	495	497	498

RHO = 0.50 SCALED INITIAL SERVER LOAD = 0.80 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH	0.30	0.40	0.50	0.60	0.80	1.00	1.50	2.00	2.50	3.0	4.0	5.0	6.0	7.0	8.0	10.0
WIENER	589	561	543	531	516	508	500	498	498	498	499	499	500	500	500	500
MD1 QUEUE	500	400	428	443	440	438	455	468	477	483	490	494	497	498	499	500
ME1 K= 4.00	500	400	414	419	423	429	447	461	471	479	488	493	496	497	498	499
ME1 K= 3.00	500	400	411	414	419	426	445	460	470	478	487	493	496	497	498	499
ME1 K= 2.00	500	400	405	407	412	421	441	457	468	476	486	492	495	497	498	499
ME1 K= 1.50	500	400	400	401	408	416	438	454	466	474	485	491	495	497	498	499
ME1 K= 1.00	500	400	394	394	401	410	433	450	463	472	484	490	494	496	498	499
ME1 K= 0.50	500	400	385	383	390	400	425	444	458	468	481	489	493	496	497	499
ME1 K= 0.20	500	400	377	374	380	391	418	438	453	464	479	487	492	495	497	499
GAMMA INPUT	500	400	369	365	371	382	411	432	449	460	476	485	491	494	496	498

RHO = 0.50 SCALED INITIAL SERVER LOAD = 0.60 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH	0.30	0.40	0.50	0.60	0.80	1.00	1.50	2.00	2.50	3.0	4.0	5.0	6.0	7.0	8.0	10.0
WIENER	463	458	457	458	462	467	478	485	490	493	496	498	499	500	500	500
MD1 QUEUE	300	341	368	385	387	411	438	456	469	478	488	493	496	498	499	499
ME1 K= 4.00	300	329	347	359	379	397	429	450	464	474	485	492	495	497	498	499
ME1 K= 3.00	300	326	342	355	376	394	427	448	463	473	485	491	495	497	498	499
ME1 K= 2.00	300	321	336	348	370	389	423	445	460	471	484	490	494	497	498	499
ME1 K= 1.50	300	317	331	343	366	385	420	443	458	469	483	490	494	496	498	499
ME1 K= 1.00	300	311	324	337	360	379	416	439	456	467	481	489	493	496	497	499
ME1 K= 0.50	300	303	314	326	350	370	408	434	451	463	479	487	492	495	497	499
ME1 K= 0.20	300	296	305	317	341	362	401	428	446	460	476	486	491	494	496	498
GAMMA INPUT	300	289	297	309	332	354	394	422	442	456	474	484	490	493	496	498

RHO = 0.50 SCALED INITIAL SERVER LOAD = 0.40 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.2	1.5	2.0	3.0	4.0	6.0
WIENER	360	351	354	360	374	388	400	411	428	442	452	464	477	490	495	499
MD1 QUEUE	350	300	250	200	256	297	325	339	361	390	404	426	449	474	486	496
ME1 K= 4.00	350	300	250	200	247	279	302	321	352	376	395	417	443	470	484	495
ME1 K= 3.00	350	300	250	200	244	275	298	318	349	374	393	415	441	469	483	494
ME1 K= 2.00	350	300	250	200	240	269	292	312	344	369	389	412	438	468	482	494
ME1 K= 1.50	350	300	250	200	236	264	288	308	340	365	386	409	436	466	481	493
ME1 K= 1.00	350	300	250	200	232	258	282	302	334	360	381	405	433	464	480	493
ME1 K= 0.50	350	300	250	200	224	250	272	292	325	352	373	398	427	460	477	492
ME1 K= 0.20	350	300	250	200	218	242	264	284	317	344	365	391	421	456	475	490
GAMMA INPUT	350	300	250	200	212	235	256	276	309	336	358	384	416	453	472	489

RHO = 0.50 SCALED INITIAL SERVER LOAD = 0.20 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.2	1.5	2.0	3.0	4.0	6.0
WIENER	211	243	269	290	324	349	369	385	410	428	442	457	473	488	494	499
MD1 QUEUE	150	100	136	172	228	269	297	313	350	377	396	420	445	473	485	495
ME1 K= 4.00	150	100	136	166	214	250	279	302	338	366	387	411	439	469	483	494
ME1 K= 3.00	150	100	135	164	211	247	275	299	335	363	385	409	437	468	482	494
ME1 K= 2.00	150	100	133	161	206	241	270	293	331	359	381	406	434	466	481	494
ME1 K= 1.50	150	100	132	159	203	238	266	289	327	355	377	403	432	464	480	493
ME1 K= 1.00	150	100	130	155	198	232	260	284	321	350	373	399	429	462	479	493
ME1 K= 0.50	150	100	126	150	191	224	252	275	313	342	365	392	423	459	476	491
ME1 K= 0.20	150	100	123	145	184	216	244	267	305	334	358	385	418	455	474	490
GAMMA INPUT	150	100	119	141	178	210	237	260	297	327	351	379	412	451	471	489

RHO = 0.50 SCALED INITIAL SERVER LOAD = 0.0 SCALED MEAN WAIT IN THOUSANDTHS

SCALED EPOCH

	0.05	0.10	0.15	0.20	0.30	0.40	0.50	0.60	0.80	1.0	1.2	1.5	2.0	3.0	4.0	6.0
WIENER	155	206	242	269	309	338	360	378	405	425	439	455	472	488	494	499
MD1 QUEUE	48	90	129	163	218	259	287	308	346	374	394	418	444	472	485	495
ME1 K= 4.00	47	88	124	155	204	242	272	296	334	363	385	410	438	468	483	494
ME1 K= 3.00	47	87	122	152	201	239	269	293	331	360	382	408	436	467	482	494
ME1 K= 2.00	46	86	120	149	197	234	263	288	327	356	378	404	433	465	481	493
ME1 K= 1.50	46	85	118	147	193	230	260	284	323	352	375	401	431	464	480	493
ME1 K= 1.00	45	83	116	143	189	225	254	279	318	347	370	397	428	462	478	492
ME1 K= 0.50	44	81	111	138	182	217	246	270	309	339	363	390	422	458	476	491
ME1 K= 0.20	43	78	108	133	175	210	238	262	301	332	356	384	417	454	473	490
GAMMA INPUT	42	76	104	129	170	203	231	255	294	324	349	377	411	450	471	489

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 85	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) TRANSIENT EFFECTS IN M/G/1 QUEUES: AN EMPIRICAL INVESTIGATION		5. TYPE OF REPORT & PERIOD COVERED Technical report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) MICHAEL R. MIDDLETON		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0418
9. PERFORMING ORGANIZATION NAME AND ADDRESS DEPT. OF OPERATIONS RESEARCH STANFORD UNIVERSITY, STANFORD, CALIF.		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-061
11. CONTROLLING OFFICE NAME AND ADDRESS OPERATIONS RESEARCH PROGRAM OFFICE OF NAVAL RESEARCH CODE 434 ARLINGTON, VA. 22217		12. REPORT DATE June 1979
		13. NUMBER OF PAGES 148
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported in part by National Science Foundation Grant ENG 75-14847 Department of Operations Research, Stanford University and issued as Technical Report No. 51 NH

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Queueing Theory; M/G/1 Queue; Transient Behavior;

Server Load; Virtual Waiting Time.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This research provides numerical results for time-dependent expected server load (mean virtual waiting time) in single-server queues with Poisson arrivals and gamma distributed service times. The results are presented in tabular form to facilitate their use by practitioners involved in the study of operating systems. next page

continued

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT (continued)

The server load process is expressed as a net input process (having stationary, independent increments) modified by a reflecting barrier at the origin. In queueing systems the net input process is compound Poisson. Two other choices of net input processes, the Wiener process (or Brownian motion) and the gamma process, provide approximations for the queueing process. This research considers groups of server load processes whose parameters are selected so that the first and second moments of their net input processes are matched. An existing Laplace transform expression is employed to obtain transient expected server load at specified epochs.

The Laplace transform is inverted numerically using an existing technique. The inversion technique is first applied to test functions whose exact inverses are known, and the results are also compared with queueing results obtained by other methods. These investigations indicate that the numerical results for expected server load have four to six significant digits when the approximation is based upon thirty four values of the Laplace transform function. The computer programs are coded in FORTRAN IV using extended double precision, and complete documentation is included.

The numerical results for expected server load are tabulated in scaled form. Each of the 93 tables includes results for a specific combination of traffic intensity parameter (twelve values, ranging from .5 through 1.0 up to 2.0) and initial server load (either six or nine values, ranging from 0 to 4 in scaled form). An individual tables has results for the Wiener process, the gamma input process, and eight queues with different service time distributions; each of the ten processes is evaluated at sixteen epochs. Step-by-step procedures for using the tables are included, and several sample problems are presented.

The tabulated results allow a comprehensive study of the error associated with using the Wiener process as an approximation of server load in queues. This study confirms that the Wiener process is always an upper bound and that the approximation is best for queues with a traffic intensity parameter near unity. The scaled results also indicate that the gamma input process and queueing process with deterministic service times provide tight lower and upper bounds, respectively, for expected server load in all queues with Poisson arrivals and gamma distributed service times.

85/51